Helsinki University of Technology Institute of Mathematics Research Reports
Espoo 2011 A597

PRECONDITIONING FOR STANDARD AND TWO-SIDED KRYLOV SUBSPACE METHODS

Mikko Byckling

Aalto University

Helsinki University of Technology Institute of Mathematics Research Reports
Espoo 2011 A597

PRECONDITIONING FOR STANDARD AND TWO-SIDED KRYLOV SUBSPACE METHODS

Mikko Byckling

Doctoral dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the Faculty of Information and Natural Sciences for public examination and debate in Auditorium K at the Aalto University School of Science and Technology (Espoo, Finland) on the 21st of January 2011 at 12 noon.

Aalto University
School of Science and Technology
Department of Mathematics and Systems Analysis

MIKKO BYCKLING
Department of Mathematics and Systems Analysis
Aalto University
P.O. Box 11100, FI-00076 Aalto, Finland
E-mail: mikko.byckling@tkk.fi

ISBN 978-952-60-3532-1 (print)
ISBN 978-952-60-3533-8 (PDF)
ISSN 1797-5867
Aalto University, Mathematics, 2011

Aalto University
School of Science and Technology
Department of Mathematics and Systems Analysis
P.O. Box 11100, FI-00076 Aalto, Finland
email: math@tkk.fi http://math.tkk.fi/

Mikko Byckling: *Preconditioning for standard and two-sided Krylov subspace methods*; Helsinki University of Technology Institute of Mathematics Research Reports A597 (2011).

Abstract: This thesis is concerned with the solution of large nonsymmetric sparse linear systems. The main focus is on iterative solution methods and preconditioning. Assuming the linear system has a special structure, a minimal residual method called TSMRES, based on a generalization of a Krylov subspace, is presented and its convergence properties studied. In numerical experiments it is shown that there are cases where the convergence speed of TSMRES is faster than that of GMRES and vice versa. The numerical implementation of TSMRES is studied and a new numerically stable formulation is presented. In addition it is shown that preconditioning general linear systems for TSMRES by splittings is feasible in some cases. The direct solution of sparse linear systems of the Hessenberg type is also studied. Finally, a new approach to compute a factorized approximate inverse of a matrix suitable for preconditioning is presented.

AMS subject classifications: 65F10,65F05,65F08

Keywords: minimal residual methods, Krylov subspaces, nonsymmetric systems, preconditioning, approximate inverses

Mikko Byckling: *Standardien ja kaksisuuntaisten Krylov-aliavaruusmenetelmien pohjustaminen*

Tiivistelmä: Tässä väitöskirjassa käsitellään suurten epäsymmetristen harvojen lineaaristen yhtälöryhmien ratkaisua. Työ käsittelee pääsääntöisesti iteratiivisiä menetelmiä ja pohjustusta. Olettaen että lineaarisella yhtälöryhmällä on erityinen rakenne, esitellään uusi minimi residuaali -menetelmä TSMRES ja tutkitaan sen konvergenssiominaisuuksia. Numeerisilla kokeilla osoitetaan, että on olemassa tapauksia joissa TSMRES konvergoi nopeammin kuin GMRES ja toisinpäin. Työssä tutkitaan myös TSMRES-menetelmän numeerista implementaatiota ja esitellään menetelmälle uusi numeerisesti stabiili muoto. Lisäksi osoitetaan, että yleinen lineaarinen yhtälöryhmä voidaan pohjustaa TSMRES-menetelmälle sopivaan muotoon jakamalla matriisi osiin. Tarkastellaan myös harvojen Hessenberg-tyyppisten lineaaristen yhtälöryhmien ratkaisua. Lopuksi esitetään lisäksi uusi pohjustukseen soveltuva menetelmä matriisin inverssin approksimatiivisen hajotelman laskentaan.

Avainsanat: minimi residuaali -menetelmät, Krylov-aliavaruudet, epäsymmetriset yhtälöryhmät, pohjustaminen, approksimatiiviset inverssit

Preface

This thesis has been written at the Department of Mathematics and Systems Analysis of the Aalto University School of Science and Technology during the years 2005-2010. Financial support has been received from the Academy of Finland and the Research Foundation of Helsinki University of Technology.

I am most grateful to my instructor Dr. Marko Huhtanen for his continuous support, guidance and seemingly abundant wealth of new ideas. I would like to thank Dr. Harri Hakula for convincing me that the field of numerical mathematics is an extremely interesting one and providing motivational discussions about mathematics and life in general. I would also like to thank Prof. Timo Eirola for supervising my thesis and for organizing very interesting study groups in numerical analysis. I am indebted to Prof. Olavi Nevanlinna, director of the institute, and all my colleagues for creating a pleasant working atmosphere. Especially I would like to thank Matti, Toni and Igor for friendship and interesting conversations about mathematics, programming, gaming and cooking.

It has been an honour to have Prof. Daniel Kressner from the ETH Zürich and Dr. Xavier Vasseur from CERFACS, two experts in the field, as preliminary examiners of the thesis. Their in depth comments about the manuscript very highly appreciated.

Finally, I would like to thank my parents and relatives for all their support during my studies. Last, but by no means least, I am greatly indebted to Mari and Neea for their love, support and understanding.

Toulouse, December 2010 Mikko Byckling

Author's contribution

Chapter 2 The two-sided minimum residual method (TSMRES) was originally introduced by Huhtanen and Nevanlinna in the article *A minimum residual algorithm for solving linear systems*, BIT, 46 (2006), pp. 533–548. All the numerical experiments comparing the standard and the two-sided Krylov subspaces and some of the theoretical results concerning TSMRES are independent research by the author.

Chapter 3 This chapter is independent research by the author. It is shown that the two-sided Krylov subspaces can be considered a special case of more general subspaces.

Chapter 4 This chapter is independent research by the author. A numerically stable version of TSMRES is introduced and the practical numerical implementation and computational complexity is considered. Theoretical considerations on the numerical stability of TSMRES are given.

Chapter 5 This chapter is independent research by the author. The convergence of the restarted TSMRES method and preconditioning general linear systems for TSMRES by splittings is considered. It is shown that there are cases where for similarly sized subspaces the restarted GMRES method stagnates whereas the restarted TSMRES converges.

Chapter 6 This chapter is independent research by the author. A new method based on the Gaussian elimination is introduced for the direct solution of sparse Hessenberg linear systems. Computational complexity and numerical stability of the new method is considered. A decomposition of a matrix into a product of a lower triangular matrix and Hessenberg matrix is introduced and its use as a preconditioner considered.

Chapter 7 This chapter is based on the article [20], i.e., M. Byckling and M. Huhtanen *Approximate factoring of the inverse*, To appear in Numerische Mathematik (2010). The original idea of Algorithms 7.1 and 7.2 to compute a factorized approximate inverse of a matrix is by Huhtanen. The author of the thesis is responsible for the practical version of the method (Algorithm 7.3), computational complexity analysis, the numerical implementation of Algorithms 7.1, 7.2 and 7.3 and all of the numerical results.

Contents

Chapter 1

Introduction

Many engineering and physical applications, such as structural analysis or the study of fluid flow, heat conduction or properties of materials are based on building an ideal mathematical model of the phenomenon. Due to structure of the problem itself or physical properties, the model may contain several interdependent unknown quantities. Then, based on the ideal model, using given initial or boundary conditions, for instance, the goal is to determine the unknowns under study.

To solve an approximation of the ideal model, discretization is often used, i.e., the problem is turned into a finite dimensional discretized model. Depending on the quantities being modelled, when solving the unknowns from the discretized model, we may finally end up solving a linear system or an eigenvalue problem.

In many applications, the final solution phase involves solving a large and *sparse* linear system, possibly consisting of hundreds of millions of unknowns. Let $A \in \mathbb{C}^{n\times n}$ be a large, sparse and nonsingular matrix. We consider solving the linear system

$$Ax = b, \tag{1.1}$$

with the right-hand side $b \in \mathbb{C}^n$ and a nonzero $x \in \mathbb{C}^n$ the unknown solution. Since A is nonsingular, the solution vector x is unique.

Another class of problems frequently appearing in applications are eigenvalue problems, defined as

$$Ax = \lambda x, \tag{1.2}$$

where $\lambda \in \mathbb{C}$ is defined as an *eigenvalue* and a nonzero $x \in \mathbb{C}^n$ the corresponding *eigenvector* of A. The eigenvalues λ are the n roots of the characteristic polynomial $\varphi(\lambda) = \det(\lambda I - A)$. The set of eigenvalues of A is called the *spectrum* and is denoted by $\sigma(A)$.

We define an n-by-n matrix to be *full* when it contains approximately $O(n^2)$ nonzero entries. Similarly, we define a n-by-n matrix to be sparse when it only contains approximately $O(\tau n)$ nonzero entries with $\tau \ll n$, i.e., most of the entries in the matrix are in fact zeroes. To take advantage of the sparsity, special care must be taken when linear systems involving sparse matrices are being solved.

Solvers for linear systems can be roughly divided into two categories: direct and iterative. Between the two approaches, there are major differences.

Direct solvers require storing all the entries in the matrix A. To solve a general linear system, direct solvers typically require from $O(n^2)$ to even $O(n^3)$ floating point operations. In addition, direct solvers require a great deal of additional storage the amount of which depends on the structure of the matrix A.

Iterative solvers only require the computation of matrix-vector products Av (or A^*v) for vectors v. When successful, typically approximately only $O(n)$ operations are required by the solution process. The amount of additional storage required by iterative solvers is usually very moderate and typically does not depend on the structure of A.

In addition to requiring more storage and floating point operations than iterative solvers, direct solvers are usually considerably more difficult to parallelize than iterative methods, which often have parallelism built in. However, unless the given problem is severely ill-conditioned, given enough time and memory, direct solvers can be expected to produce a solution to the given problem. Iterative solvers, on the other hand, do not necessarily converge to a solution at all.

In this thesis, new methods for solving sparse linear systems, especially ones with some predefined structure, are studied. The emphasis will be on iterative methods, although some consideration will be given also for sparse direct solvers. Most of the analysis will be given assuming exact arithmetic. The effect of using finite precision arithmetic can not be completely avoided in practice, however. Whenever needed, we will give alternative formulations of the proposed algorithms for which properties of using finite precision arithmetic have been taken into account.

In the following sections, we give a short introduction to solving general linear systems with references to literature. We consider both direct and iterative solvers for sparse linear systems. For iterative solvers, we present some important results and concepts which will be referred to in the following chapters. Since it is so extremely important in practice, we give a short introduction to preconditioning linear systems for iterative solvers. Finally, at the end of this chapter, we give a short outline of the contents of this thesis.

1.1 Direct methods

Direct solution of general sparse linear systems (1.1) is based on the Gaussian elimination. There exists numerous textbooks on the subject, see for instance Duff, Erisman and Reid [36] or a recent implementation oriented approach by Davis [29]. We begin by considering the case where the matrix A is full.

Consider a factorization of a nonsingular full n-by-n matrix A into two n-by-n matrices,

$$A = LU, \tag{1.3}$$

where L is a lower triangular and U is an upper triangular n-by-n matrix. Factorization of the form (1.3) is called an *LU-decomposition*. On how L and U are constructed by using the Gaussian elimination, see any textbook on linear algebra, such as [55] or [117]. For a full matrix, the floating point cost of computing (1.3) is approximately $O(\frac{2}{3}n^3)$ operations.

The decomposition (1.3) does not always exist even though A is nonsingular. The remedy is to apply *partial pivoting*

$$PA = LU, \tag{1.4}$$

where L and U are as before and P denotes a permutation matrix. In exact arithmetic, a decomposition of the form (1.4) always exists when A is nonsingular [132]. In finite precision arithmetic, P is usually chosen to minimize numerical error, see for instance [55, 61]. In most practical cases, partial pivoting yields a numerically stable decomposition of the form (1.4) [61].

For simplicity, in the following we consider only the case without partial pivoting. With the decomposition (1.3), a linear system $Ax = b$ is readily solvable. The solution $x = U^{-1}L^{-1}b$ is computed in two steps: first by solving a lower triangular linear system and then an upper triangular one. We then have

$$Ly = b, \quad Ux = y. \tag{1.5}$$

The formulation (1.5) is effective because the solution of upper triangular linear systems $Ux = y$ (and similarly that of lower triangular systems $Ly = b$) is inexpensive in terms of floating point operations. Denote the jth component of vectors x and b by x_j and b_j, respectively. Also, let $u_{j,k}$ denote the element at position (j, k) in the matrix U. Then, $x_n = b_n/u_{n,n}$ and for $j = n-1, \ldots, 1$,

$$x_j = (b_j - \sum_{k=j+1}^{n} u_{j,k}x_k)/u_{j,j}, \tag{1.6}$$

with a total floating point cost of approximately $O(n^2)$ operations.

One of the advantages of using an LU-decomposition is that the factors L and U can be efficiently reused to solve the same linear system with a different right-hand side. When an LU-decomposition has been computed for a given matrix A, the solution of the linear system involving A requires only $O(2n^2)$ floating point operations. Thus, if the same linear system needs to be solved for multiple right-hand sides, the initial factorization cost $O(\frac{2}{3}n^3)$ can be partially amortized among the different solves.

Now consider the case where an LU-decomposition is used to solve the sparse linear system (1.1). Then, even though the matrix A is sparse, the computed factors are neccessarily not. A classical example is to consider computing an LU-decomposition of a matrix having the sparsity pattern

$$A = \begin{bmatrix} \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & & & \\ \mathrm{x} & & \mathrm{x} & & \\ \mathrm{x} & & & \mathrm{x} & \\ \mathrm{x} & & & & \mathrm{x} \end{bmatrix}, \tag{1.7}$$

where x denotes a nonzero element. Then, after computing the LU-decomposition, we have the factors as

$$L = \begin{bmatrix} \mathrm{x} & & & & \\ \mathrm{x} & \mathrm{x} & & & \\ \mathrm{x} & \mathrm{o} & \mathrm{x} & & \\ \mathrm{x} & \mathrm{o} & \mathrm{o} & \mathrm{x} & \\ \mathrm{x} & \mathrm{o} & \mathrm{o} & \mathrm{o} & \mathrm{x} \end{bmatrix}, U = \begin{bmatrix} \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ & \mathrm{x} & \mathrm{o} & \mathrm{o} & \mathrm{o} \\ & & \mathrm{x} & \mathrm{o} & \mathrm{o} \\ & & & \mathrm{x} & \mathrm{o} \\ & & & & \mathrm{x} \end{bmatrix}, \tag{1.8}$$

where o denotes an extra nonzero element produced by the triangularization process. The extra nonzero elements appearing in the factors L and U are usually called *fill-in*.

To reduce the fill-in, a reordering of the unknowns is often performed. For a matrix having the sparsity pattern (1.7), we have

$$PAQ = \begin{bmatrix} \mathrm{x} & & & & \mathrm{x} \\ & \mathrm{x} & & & \mathrm{x} \\ & & \mathrm{x} & & \mathrm{x} \\ & & & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \end{bmatrix}, \tag{1.9}$$

where P and Q denote permutation matrices which reverse the order of rows and columns. For the factors L and U of the permuted matrix PAQ we then

have

$$L = \begin{bmatrix} \mathrm{x} & & & & \\ & \mathrm{x} & & & \\ & & \mathrm{x} & & \\ & & & \mathrm{x} & \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \end{bmatrix}, U = \begin{bmatrix} \mathrm{x} & & & & \mathrm{x} \\ & \mathrm{x} & & & \mathrm{x} \\ & & \mathrm{x} & & \mathrm{x} \\ & & & \mathrm{x} & \mathrm{x} \\ & & & & \mathrm{x} \end{bmatrix}, \tag{1.10}$$

i.e., no fill-in occurs and the sparsity patterns of the matrices L and U equal to those of the lower and upper triangular parts of the matrix PAQ.

To minimize the fill-in, different types of reorderings are commonly used with sparse direct solvers. Types of fill-in minimizing reorderings include, for instance, Reverse Cuthill-McKee [51] and Approximate Minimum Degree [30]. The study of fill-in minimizing reorderings is tightly linked with the graph theory of the underlying connectivity graph of the matrix, see for example [32] and [130].

A catastrophic fill-in such as that described in (1.7) rarely happens in practice. Thus, for sparse linear systems, complexity of the factorization is rarely of the order $O(n^3)$. Similarly, in spite of the fill-in, solution of a sparse lower and upper triangular systems related to the computed factors L and U can be computationally considerably less expensive than that of the full ones. However, for very large linear systems, especially for those arising from discretizations of three-dimensional problems, time and memory consumption remain an problem. Another problem with the LU-decomposition is that its computation is inherently sequential and therefore very difficult to parallelize.

There exists several software packages for sparse direct solvers. These include, for instance, UMFPACK [28], SuperLU [31] and MUMPS [2]. Such software is often highly complex, consisting of tens of thousands of lines of code. To achieve maximum efficiency on a modern computer, sophisticated implementation techniques such as blocking [31] and parallel graph partitioning [22, 76] are used. Although efficient parallelization of direct solvers is still problematic, it has gained advances in recent years, see for instance [1, 57].

1.2 Iterative methods

Iterative methods based on Krylov subspaces are among the most important techniques for solving large sparse linear systems. With the increased use of three-dimensional models and massively parallel computing architectures [115], the handicaps of direct solvers, i.e., increased factorization cost and increased memory usage due to fill-in and difficulties in parallellization, become

more apparent. Due to such constraints, application of iterative methods in large industrial problems is nearly standard.

Iterative methods have been studied actively since the 1950's. The idea of solving linear systems iteratively has been around much longer, and dates back to Gauss in the 1820's or even further [125]. The results of papers now viewed as foundational such as those of Lanczos [80] as well as Hestenes and Stiefel [60], were initially disappointing since the methods described were considered as direct solution methods and thus did not compare favourably with the existing solution techniques.

In the 1970's it was realised that iterative solvers could be used to approximate the solution of a large and sparse linear system, see for instance [97]. Ideas such as preconditioning were shown to be applicable to the solution of large linear systems. Even though the research into iterative methods has been an active area ever since, many questions still remain unanswered [54, 107, 110, 122].

Iterative methods are nowadays well covered in literature. Books of Saad [105] and van der Vorst [121], for instance, provide complete overviews on modern iterative solution techniques. Software implementations of iterative methods are also widely available, see for instance [47, 104].

Mathematically most of the modern iterative methods are based on projection processes. Let x_m be an approximate solution of (1.1) from a subspace $\mathcal{K}_m$ of dimension m. To select x_m, we apply a Petrov-Galerkin condition to the residual r_m. We then have

$$r_m = b - Ax_m \ \perp \mathcal{L}_m, \tag{1.11}$$

where $\mathcal{L}_m$ denotes another subspace of dimension m. Traditionally the subspaces $\mathcal{K}_m$ and $\mathcal{L}_m$ are selected as Krylov subspaces of the matrix A, described in what follows.

1.2.1 Krylov subspaces

Denote by $q_1 \in \mathbb{C}^n$ a vector with $||q_1||_2 = 1$. Let $\mathcal{K}_j = \mathcal{K}_j(A; q_1)$ denote a Krylov subspace, defined as

$$\mathcal{K}_j(A; q_1) = \underset{p \in \mathbb{P}_{j-1}}{\text{span}} \{p(A)q_1\} = \text{span}\{q_1, Aq_1, \ldots, A^{j-1}q_1\}, \tag{1.12}$$

where $\mathbb{P}_j$ denotes a set of polynomials of degree j at most. As expected, different choices of subspaces $\mathcal{K}_m$ and $\mathcal{L}_m$ yield different Krylov subspace methods.

Algorithm 1.1 describes a method for constructing an orthonormal basis $\{q_1, q_2, \ldots, q_m\}$ for the Krylov subspace $\mathcal{K}_m(A; q_1)$.

Algorithm 1.1 Arnoldi method

Choose a vector q_1 with $||q_1||_2 = 1$
for $j = 1, 2, \ldots, m$ **do**
    $h_{l,j} = (Aq_j, q_l), \quad l = 1, \ldots, j$
    $w_j = Aq_j - \sum_{l=1}^{j} h_{l,j} q_l$
    $h_{j+1,j} = ||w_j||_2$
    if $h_{j+1,j} = 0$ **then**
        return
    end if
    $q_{j+1} = w_j / h_{j+1,j}$
end for

Algorithm 1.1, usually called the Arnoldi method, was originally introduced by Arnoldi in [3]. It uses a standard Gram-Schmidt process for the construction of an orthonormal set of basis vectors for the Krylov subspace $\mathcal{K}_m(A; q_1)$. Another formulation of the Arnoldi method is given in Algorithm 1.2.

Algorithm 1.2 Arnoldi Modified Gram-Schmidt method

Choose a vector q_1 with $||q_1||_2 = 1$
for $j = 1, 2, \ldots, m$ **do**
    $w_j = Aq_j$
    for $l = 1, \ldots, j$ **do**
        $h_{l,j} = (w_j, q_l)$, $w_j = w_j - h_{l,j} q_l$
    end for
    $h_{j+1,j} = ||w_j||_2$
    if $h_{j+1,j} = 0$ **then**
        return
    end if
    $q_{j+1} = w_j / h_{j+1,j}$
end for

In Algorithm 1.2, the modified Gram-Schmidt process has been used to construct an orthonormal set of basis vectors for $\mathcal{K}_m(A; q_1)$. Mathematically Algorithms 1.1 and 1.2 are equivalent, but in finite precision arithmetic the modified Gram-Schmidt implementation is numerically much more stable; see Björck [14].

In a parallel computing environment the problem with Algorithm 1.2 is its sequential nature. Algorithm 1.1, on the other hand, is very attractive for parallel computations. It was shown by Giraud et al. in [53], that a

numerical stability may be regained for the standard Gram-Schmidt process by performing a second orthogonalization step for the generated basis vectors. Then, although the second orthogonalization step effectively doubles the amount of computations required, the use of the standard Gram-Schmidt process in a parallel environment is feasible.

Denote by $Q_m = [q_1\ q_2\ \cdots\ q_m]$ the matrix having the generated orthonormal basis vectors q_j as its columns. Also, denote by $\bar{H}_m$ the $(m+1)$-by-m matrix with entries $h_{i,j}$ determined by Algorithm 1.1. Assuming that the Algorithm 1.1 does not break down before the mth step, we have the relation

$$AQ_m = Q_m H_m + w_m e_m^T = Q_{m+1}\bar{H}_m, \tag{1.13}$$

where e_m denotes the mth column of an m-by-m identity matrix and H_m denotes the m-by-m matrix obtained after the last row of $\bar{H}_m$ has been removed.

The minimal polynomial of the matrix A at q_1 is the monic polynomial $\wp$ of the least degree satisfying $\wp(A)q_1 = 0$. If the Arnoldi process breaks down at step j, we have generated the minimal polynomial of q_1 and an invariant subspace for A as stated by the following proposition [105, Proposition 6.6, pp. 155].

Proposition 1.1. *Arnoldi's method, i.e., Algorithm 1.1 breaks down at step j if and only if the minimal polynomial of q_1 is of degree j. Moreover, in this case the subspace $\mathcal{K}_j$ is invariant under A.*

In addition to computing a Krylov subspace $\mathcal{K}_m(A; q_1)$, the Arnoldi method can be used to approximate the eigenvalues of a large and sparse matrix. By the relation (1.13), we have a decomposition of the matrix A as $H_m = Q_m^* A Q_m$. In view of the eigenvalue problem (1.2), consider

$$H_m g = \theta g, \tag{1.14}$$

with $\theta \in \mathbb{C}$ and a nonzero $g \in \mathbb{C}^m$. Here the pair $(\theta, Q_m g)$ is usually called a Ritz pair. Once the eigenvalues θ_j and eigenvectors g_j of the problem (1.14) have been computed for $j = 1, \ldots, m$, the pairs $(\theta_j, Q_m g_j)$ can be used to approximate the eigenvalues and eigenvectors of the matrix A.

In the general case it is still a tough question how the eigenvalue estimates produced by the Arnoldi method relate to the true eigenvalues of A. For interesting results see [6, 43, 101, 122] and references therein.

1.2.2 GMRES - Generalized Minimal Residual Method

The generalized minimal residual method (GMRES) was introduced by Saad and Schultz in [106]. GMRES can be considered as a projection method

on a Krylov subspace. Denote by x_0 an initial approximate solution. With the equation (1.11) we then have the GMRES method by setting $\mathcal{K}_m = \mathcal{K}_m(A; q_1)$, $\mathcal{L}_m = A\mathcal{K}_m$ and $q_1 = r_0/||r_0||_2$, where $r_0 = b - Ax_0$ denotes the initial residual.

At the mth step of iteration, a correction z_m for the iterate $x_m = x_0 + z_m$ is determined in the Krylov subspace $\mathcal{K}_m(A; q_1)$ by solving the least-squares problem

$$\min_{v \in \mathcal{K}_m(A;q_1)} ||b - A(x_0 + v)||_2. \tag{1.15}$$

By using (1.13), we then have

$$\min_{v \in \mathcal{K}_m(A;q_1)} ||b - A(x_0 + v)||_2 = \min_{y_m \in \mathbb{C}^m} ||\beta e_1 - \bar{H}_m y_m||_2, \tag{1.16}$$

where $\bar{H}_m$ as before, $\beta = ||r_0||_2$ and e_1 denotes the first column of an $(m+1)$-by-$(m+1)$ identity matrix.

To find the minimizer y_m, we need to solve a $(m+1)$-by-m least-squares problem. Since m is typically small, finding y_m is computationally inexpensive. Once the minimizer has been computed, we have the approximate solution as $x_m = x_0 + z_m = x_0 + Q_m y_m$.

With these, we have Algorithm 1.3, which uses a modified Gram-Schmidt process to construct an orthonormal basis for $\mathcal{K}_m(A; q_1)$.

Algorithm 1.3 GMRES

1: $r_0 = b - Ax_0$, $\beta = ||r_0||_2$, $q_1 = r_0/\beta$
2: **for** $j = 1, 2, \ldots, m$ **do**
3: $\quad w = Aq_j$
4: $\quad$ **for** $l = 1, \ldots, j$ **do**
5: $\quad\quad h_{l,j} = (w, q_l)$, $w = w - h_{l,j} q_l$
6: $\quad$ **end for**
7: $\quad h_{j+1,j} = ||w||_2$
8: $\quad$ **if** $h_{j+1,j} = 0$ **then**
9: $\quad\quad$ Set $m = j$ and **goto** 12
10: $\quad$ **end if**
11: $\quad q_{j+1} = w/h_{j+1,j}$
12: $\quad$ Compute $y_m = \text{Argmin}_y ||\beta e_1 - \bar{H}_m y||_2$
13: $\quad x_m = x_0 + Q_m y_m$
14: **end for**

We note that since the GMRES method minimizes the residual norm in $\mathcal{K}_m(A; q_1)$, it is optimal, i.e., no other iterative method can produce a smaller

residual norm over the same subspace. Due to the minimization property, for the residual norms of the GMRES method, it holds

$$||r_m||_2 \leq ||r_{m-1}||_2 \leq \cdots \leq ||r_0||_2,$$

i.e., the residual norm is always nonincreasing.

In exact arithmetic, GMRES produces an exact solution in at most n iterations. In finite precision arithmetic, GMRES based on a modified Gram-Schmidt process is backward stable [94], that is, after n steps the approximation produced can be expected to be numerically accurate unless the problem is severely ill-conditioned.

When the Arnoldi process used by GMRES breaks down, the solution produced by GMRES is exact. The following theorem is from Saad [105, Proposition 6.10, pp.171].

Theorem 1.2. *Let A be a nonsingular matrix. Then the* GMRES *algorithm breaks down at step j, i.e., $h_{j+1,j} = 0$, if and only if the approximate solution x_j is exact.*

An equivalent condition for the breakdown of GMRES is that the degree of the minimal polynomial of A at the initial residual vector r_0 is equal to j, see [106].

In the general case, it is very difficult to say anything about the iterative behaviour of the residual norm of GMRES. Unless the spectrum of A completely surrounds the origin, the polynomial approximation used by GMRES can be expected to be successful [91], see also [33]. However, in the general case it is possible that GMRES produces no reduction of the residual before the nth step [56].

When A is diagonalizable as $A = XDX^{-1}$, it readily follows that [105, Proposition 6.32, pp. 206],

$$\min_{v \in \mathcal{K}_m(A;q_1)} ||b - Av||_2 \leq \min_{\uparrow \in \mathbb{P}_m} \max_{\lambda \in \sigma(A)} |\uparrow(\lambda)| \; \kappa(X)||r_0||_2, \tag{1.17}$$

where $\mathbb{P}_j$ denotes the set of monic polynomials of degree j at most, $\kappa(X) = ||X||_2||X^{-1}||_2$ the condition number of X and λ an eigenvalue of A. When the matrix X is not too ill-conditioned, inequality (1.17) roughly states that the convergence of GMRES is determined by a polynomial approximation problem on the spectrum of A.

When the matrix A is not diagonalizable or the matrix X is very ill-conditioned, bounds of the type (1.17) are of little value. In such cases, it may be of use to link the convergence of GMRES to the ε-pseudospectra

of A, defined as $\sigma_\varepsilon = \{z \in \mathbb{C} \| \ ||(zI - A)^{-1}||_2 \geq \varepsilon^{-1}\}$, with the convention $||(zI - A)^{-1}||_2 = \infty$ when $z \in \sigma(A)$ [116]. Then, it follows that

$$\min_{v \in \mathcal{K}_m(A;q_1)} ||b - Av||_2 \leq \min_{\curlywedge \in \mathbb{P}_m} \max_{\lambda \in \sigma_\varepsilon(A)} |\curlywedge(\lambda)| \ \frac{L}{2\pi\varepsilon} ||r_0||_2, \tag{1.18}$$

where L denotes the arc length of the boundary $\partial\sigma_\varepsilon(A)$ [90]. By inequality (1.18), we can again roughly state that the convergence of GMRES is determined by a polynomial approximation problem on the pseudospectra of A.

We now consider the computation complexity of GMRES. The Arnoldi process used by GMRES requires approximately $O(2m^2n)$ operations for the orthogonalizations plus m matrix vector products to construct a Krylov subspace $\mathcal{K}_m(A; q_1)$. For a subspace of size $m+1$, the amount of storage required is $O((m+1)n)$. Since generally $m \ll n$, the operation counts for computing the minimizer y_m and computing the approximate solution x_m are negligible. Due to the growth in both the number of operations required for orthogonalizations and in the amount of memory consumed, the full GMRES method is not practical for large m.

The simplest remedy is to restart the method with $x_0 = x_m$ once the size of the subspace has reached some predefined value m. Then the excessive growth in the number of operations required and the amount of memory consumed is avoided, but the optimality of the algorithm is lost. When the method is restarted, the reduction of the residual norm per iteration step is usually reduced when compared to the unrestarted method. What is even more severe is that the restarted GMRES method can stagnate when the matrix A is not positive definite, i.e., no reduction of the residual norm is achieved after a restart, see [42, 105].

To have a method which behaves more like the full GMRES, several remedies have been suggested. These include using a truncated version of the recurrence (DQGMRES) [108], augmenting the Krylov subspace with spectral information obtained before the restart (GMRES-DR, GMRES-IR) [87, 88, 89] or using a nested Krylov subspace method (GMRESR [123],FGMRES [102]), see also [40].

1.2.3 Other iterative methods

Another large class of Krylov subspace methods is obtained with the choice $\mathcal{K}_m = \mathcal{K}_m(A; r_0)$ and $\mathcal{L}_m = \mathcal{K}_m(A^T; r_0)$ in the equation (1.11). Most of the methods in this class are based on the biorthogonalization algorithm of Lanczos [80].

Let the matrices $V_m = [v_1 \ \cdots \ v_m]$ and $W_m = [w_1 \ \cdots \ w_m]$ contain the basic vectors spanning $\mathcal{K}_m(A; v_1)$ and $\mathcal{K}_m(A^T; w_1)$ as their columns, respectively. In addition, we assume the vectors v_j and w_j are biorthogonal such that

$$W_m^T V_m = D_m, \tag{1.19}$$

where D_m denotes an m-by-m diagonal matrix, holds. The Lanczos biorthogonalization process constructs a relation

$$W_m^T A V_m = T_m, \tag{1.20}$$

where T_m denotes a tridiagonal matrix. For the details on how the relation is (1.20) is constructed, see for example [105, Chapter 7.1, pp. 217].

The difficulty in achieving the relation (1.20) is that the construction can suffer a breakdown if either one of the vectors v_j or w_j cannot be constructed or if their inner product becomes zero even though both v_j and w_j are nonzero. In the first case, with $v_j = 0$, the subspace $\mathcal{K}_{j-1}(A; v_1)$ is invariant and the solution is exact, similarly to Theorem 1.2. The other two cases are more serious and require techniques such as look-ahead in order for the construction to continue, see [95].

Based on the Lanczos biorthogonalization process, several algorithms have been proposed for the solution of linear systems. The most well-known of these are BiCG [46] and QMR [49].

In some applications operations with the matrix A^T can be difficult to implement. The operations on A^T also do not directly contribute to the solution of the linear system (1.1), hence making the solution process inefficient. To avoid the computations with A^T, several transpose-free methods such as BiCGStab [120] and TFQMR [48] have been proposed. In practice such methods can be very efficient, although there are few theoretical results concerning their convergence.

There exists also modern iterative methods for solving linear systems not based on Krylov subspaces. An important class of such methods are multigrid (MG) methods. The MG methods are used mainly for linear systems arising from partial differential equations and are largely problem specific, i.e., they exploit the knowledge of the underlying equations in the solution process. The MG methods can be very efficient. In the case of elliptic partial differential equations, for instance, the computational complexity scales linearly related to the problem size. Also, in most cases the MG methods can be efficiently parallelized. For an introduction to the subject see Multigrid tutorial [18] and references therein.

1.2.4 Preconditioning iterative methods

> *Finding a good preconditioner to solve a given sparse linear system is often viewed as a combination of art and science.*
>
> –Yousef Saad, Iterative Methods for Sparse Linear Systems (2003)

In large-scale computations, if an iterative method is applied directly to the linear system (1.1), the convergence speed attained is often too slow to be of any practical value. By the convergence analysis of the GMRES method, this may happen, for instance, when the spectrum of A nearly encloses the origin.

Efficiency of the iterative methods based on Krylov subspaces can be improved by transforming the original linear system to a one that has the same solution, but is easier to solve with an iterative method. This kind of a transformation, called *preconditioning*, is extremely important when iterative methods are used for solving industrial-scale problems. In fact, when a good preconditioner is used, the performance of many commonly used iterative solvers becomes nearly equivalent [7].

In the following, we denote a preconditioner operator by P. As with the matrix A, with the operator P we only need to be able to compute matrix-vector products of the form Pv. With the linear system (1.1), we can apply the preconditioner P from the left or from the right. A formal multiplication from the left with P yields a left preconditioned linear system

$$PAx = Pb. \tag{1.21}$$

Similarly, after a change of variables we have a right preconditioned linear system as

$$APy = b, \quad x = Py. \tag{1.22}$$

With GMRES, the left and right preconditioned versions may seem similar, but different residual norms are actually minimized. GMRES applied to the linear system (1.21) minimizes $||P(b - Ax_m)||_2$, whereas GMRES applied to the linear system (1.22) minimizes $||b - Ax_m||_2$ [105]. With the right preconditioning, flexible variants of the GMRES method, i.e., methods which allow the preconditioner to change each step, may be developed [40, 102, 123].

In a rough sense, the preconditioner operator Pv needs to approximate the product $A^{-1}v$ to be effective. On the other hand, the operations on P must be inexpensive to compute. The two requirements are often in conflict, and therefore finding a good preconditioner may be challenging.

In the following, we consider preconditioning by incomplete factorizations and approximate inverses in more detail. Another commonly used technique

is to use a basic iterative method, such as SOR, SSOR [135] or ADI [125], as a preconditioner. When a flexible Krylov subspace method such as FGMRES or GMRESR is used, another Krylov subspace method such as the restarted GMRES can be used as a preconditioner. Also MG methods can be used as preconditioners [7]. For other types of preconditioners and a more complete list of references, see [7] and references therein.

An incomplete LU-decomposition (ILU) is one of the most commonly used and effective preconditioning techniques. By computing the LU-decomposition approximately, the factorization and memory costs can be reduced. We then have

$$A \approx \bar{L}\bar{U}, \tag{1.23}$$

where the ILU-factors $\bar{L}$ and $\bar{U}$ denote sparse approximations of the actual LU-factors L and U of the matrix A. With the ILU-decomposition (1.23), we have the preconditioner as $A^{-1} \approx P = \bar{U}^{-1}\bar{L}^{-1}$. As with the standard LU-decomposition, an efficient parallelization of the ILU-decomposition may be difficult due to the serial nature of the factorization process.

The sparsity of the ILU-factors is usually ensured by either selecting the sparsity structures for $\bar{L}$ and $\bar{U}$ beforehand or by constructing the factors by using numerical dropping, i.e., discarding the small entries as the LU-decomposition is being computed. Compared to the standard LU-decomposition, not even pivoting ensures the nonsingularity of the computed ILU-factors $\bar{L}$ and $\bar{U}$ in the general case. Also, the location and magnitude of the nonzero entries of the factors tends to have an effect on the quality of the computed preconditioner.

The sparsity structures of the factors $\bar{L}$ and $\bar{U}$ can be chosen to match those of the lower -and upper triangular parts of A. Since no fill-in occurs, this technique is called ILU(0) [85]. A straightforward extension of ILU(0) is ILU(k), where k levels of fill-in are allowed to occur in the factorization process [105].

A more sophisticated ILU-decomposition is acquired by constructing the sparsity patterns dynamically by using numerical dropping when the ILU-factors are computed. In ILUT [103], the entries are discarded during the factorization process if they are deemed numerically small compared to the other entries of the computed factorization. The memory consumption is controlled by allowing only a certain number of entries in each column of the computed factors.

Both ILU(k) and ILUT can produce factors which are either singular or too ill-conditioned to be of any practical use. It has been shown in [16] and [82], that the robustness of the ILU-decomposition is increased when the numerical dropping is performed by estimating the norms of the inverses of

the factors $\bar{L}$ and $\bar{U}$.

With the ILU-decomposition, the preconditioner used to approximate A^{-1} is constructed from a factorization which approximates the matrix A. Another approach is to construct a preconditioner by approximating the matrix A^{-1} directly. The approximate inverse minimization problem is defined as

$$\min_{P \in \mathcal{W}} ||I - AP||_F \tag{1.24}$$

where $\mathcal{W}$ denotes some sparsity structure. The minimization problem (1.24) readily decouples into a set of columnwise minimization problems

$$\sum_{j=1}^{n} \min_{p_j \in \mathcal{W}_j} ||e_j - Ap_j||_2^2, \tag{1.25}$$

where $\mathcal{W}_j$ denotes the sparsity structure of the jth column and e_j and p_j denote the jth columns of n-by-n identity matrix and the preconditioner P, respectively. Applying the computed approximate inverse preconditioner P requires only a computation of a matrix-vector product and is thus readily parallelizable.

As with the ILU-decomposition, the sparsity structure $\mathcal{W}$ can be determined adaptively by using numerical dropping. Such an approach has been proposed, for instance, when the minimization problems (1.24) and (1.25) are solved with the minimum residual iteration [25]. In the SPAI method, proposed by Grote and Huckle in [58], the sparsity structure $\mathcal{W}$ is selected optimally. This can be computationally expensive even in a parallel environment [9]. Another way to select sparsity patterns of good quality is to compute them a-priori by using the powers of A and numerical dropping [24].

Also factorized approaches to computing approximate inverses of general matrices have been suggested. In [10], Benzi and Tůma proposed a method based on the biconjugation process for the construction of a factorized approximate inverse. Assume that the matrix A has a decomposition $A = LDU$, where L and U denote the unit lower -and upper triangular matrices, respectively. Then the inverse of A can be approximated as

$$A^{-1} \approx P = \bar{Z} D^{-1} \bar{W}, \tag{1.26}$$

with $\bar{Z} \approx U^{-1}$ and $\bar{W} \approx L^{-1}$. Even though the process for computing the decomposition (1.26) is partially sequential, the computation may be parallelized with graph partitioning techniques [11]. For another recent approximate factored inverse approach, see [19].

A preconditioner based on either the ILU-decomposition or the approximate inverse can benefit from the scaling and reordering of the matrix. Traditionally symmetric reorderings such as Reverse Cuthill-McKee [51] have

been known to be efficient in the context of the ILU-decompositions [105]. For nonsymmetric and highly nonnormal problems, the use of nonsymmetric reorderings, such as those presented in [37], can lead to noticeable improvements in the robustness and the quality of the acquired preconditioners [8]. For an overview on the reordering techniques and their uses in preconditioning, see [7] and references therein.

As with iterative methods, these exists several software libraries for the computation of preconditioners. These include, for instance ILUPACK for ILU-decompositions [17] and ParaSails for approximate inverses [23].

1.3 Thesis outline

The rest of the thesis is organized as follows. In Chapter 2 we consider extending the notion of a Krylov subspace assuming the linear system has a special structure. Based on the extended subspaces, we introduce a new iterative method, called the two-sided minimum residual method or TSMRES, for solving linear systems with the assumed special structure. With numerical experiments, we then analyse the TSMRES method and relate its convergence to that of GMRES.

In Chapter 3, we theoretically consider further generalizations of Krylov subspaces and their uses. Chapter 4 is mostly concerned with the practical aspects such as the implementation, complexity and numerical stability of TSMRES. In Chapter 4, we also present a numerically stabilized implementation of TSMRES.

Restarting and preconditioning of the TSMRES method is considered in Chapter 5. It is shown that compared to the restarted GMRES, the restarted TSMRES is somewhat more robust, especially when the size of the subspace is small. We also present several ways to transform a general linear system to the form required by TSMRES and make numerical comparisons of convergence with GMRES on the transformed systems. Related to the preconditioning linear systems for TSMRES, in Chapter 6 we consider a sparse direct solver for sparse linear systems of the Hessenberg form.

Finally, in Chapter 7, we consider the preconditioning of a general sparse linear system (1.1) by a factored approximate inverse. To this end, we convert the approximate inverse factorization problem to a nullspace approximation problem and present an algorithm based on the power method for computing the factors. To conclude, we present numerical experiments displaying the effectiveness of our factorized approximate inverse approach.

Chapter 2

TSMRES - Two-Sided Minimal Residual Method

2.1 Introduction

In this chapter, we consider solving a linear system

$$(I + S)x = b, \tag{2.1}$$

with $b \in \mathbb{C}^n$ and $S \in \mathbb{C}^{n\times n}$ a matrix that is readily invertible, i.e., solving linear systems with the matrix S is inexpensive in terms of floating point operations. Linear systems of this form appear often in practice, typically through splitting the original matrix. For more examples of linear systems having the form (2.1), see Section 2.4. When solving (2.1) with traditional Krylov methods, such as GMRES, we generate a subspace containing polynomials in S (or respectively S^{-1}) applied to b. In that case, however, we do not fully benefit from the structure of (2.1).

In [71] Huhtanen and Nevanlinna described an iterative method for solving (2.1), which takes into account that the matrix S is readily invertible. Let x_0 denote the initial approximate solution and $r_0 = b - (I + S)x_0$ the initial residual. Then, to solve (2.1), we set $q_1 = r_0/||r_0||_2$ and construct two-sided Krylov subspaces of the form

$$\mathcal{K}_j^{\pm}(S; q_1) = \underset{\mathfrak{p},\mathfrak{q}\in\mathbb{P}_j}{\text{span}} \{\mathfrak{p}(S)q_1, \mathfrak{q}(S^{-1})q_1\}, \tag{2.2}$$

where $\mathbb{P}_j$ denotes the set of polynomials of degree j at most. By using a Gram-Schmidt–type process, we describe how to construct an orthonormal basis for the two-sided Krylov subspace $\mathcal{K}_j^{\pm}(S; q_1)$. We then have

$$SQ_{2j+1} = Q_{2j+2}\bar{H}_{2j+1},$$

where $Q_{2j+1} \in \mathbb{C}^{n\times(2j+1)}$ has orthonormal columns spanning (2.2) and $\bar{H}_j \in \mathbb{C}^{(2j+2)\times(2j+1)}$ has a Hessenberg-like structure.

With the two-sided Krylov subspaces $\mathcal{K}_j^{\pm}(S; q_1)$, we describe a minimum residual approach for approximately solving (2.1). Denote by $\lceil\cdot\rceil$ ceiling to an integer. At the mth step of iteration, m odd, a correction z_m for the iterate $x_m = x_0 + z_m$ is determined in the two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil\frac{m}{2}\rceil}(S; q_1)$ by solving the least-squares problem

$$\min_{v\in\mathcal{K}^{\pm}_{\lceil\frac{m}{2}\rceil}(S;q_1)} ||b - (I+S)(x_0+v)||_2. \tag{2.3}$$

In this sense, the resulting iteration, called the two-sided minimal residual (TSMRES) method, is optimal in $\mathcal{K}^{\pm}_{\lceil\frac{m}{2}\rceil}(S; q_1)$. The proposed minimum residual approach is similar to the one used in the generalized minimal residual (GMRES) method of Saad and Schultz [106], except that the standard Krylov subspace $\mathcal{K}_{m+2}(S; q_1)$, based on polynomials in S (or S^{-1}), is replaced with the two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil\frac{m}{2}\rceil}(S; q_1)$, based on polynomials in both S and S^{-1}.

In our approach, the polynomials $\mathfrak{p}$ and $\mathfrak{q}$ in S and S^{-1} in the two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil\frac{m}{2}\rceil}(S; q_1)$ are always chosen to be of equal degree. We note that there are no theoretical obstructions to choose $\mathfrak{p}$ and $\mathfrak{q}$ to be of different degree. Such an approach could be applied in the cases where, for instance, operations with S^{-1} are computationally significantly more demanding than operations with S.

Due to rounding errors, numerical computation of the two-sided Krylov subspaces is a delicate matter. We demonstrate that without modifications, the implementation for the construction given in [71] is not numerically stable in certain cases. For clarity, we postpone the presentation of the numerically stable implementation of TSMRES to Chapter 4. In particular, all the numerical examples presented in this chapter have been computed with the numerically stabilized version of TSMRES.

After the derivation, we analyze the properties of the TSMRES method and the constructed two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil\frac{m}{2}\rceil}(S; q_1)$. Assuming exact arithmetic, we show that analogously to GMRES, a breakdown in the construction of the two-sided subspace $\mathcal{K}^{\pm}_{\lceil\frac{m}{2}\rceil}(S; q_1)$ implies that the generated solution x_m is exact. Thus, the exact solution x_n is reached in $\lceil\frac{n}{2}\rceil$ iterations for a subspace of size n at most.

Finally, at the end of this chapter, we numerically analyze the convergence properties of TSMRES. It is shown that there are cases where convergence of the TSMRES method is faster than that of the standard GMRES method and vice versa.

In the context of solving large-scale Lyapunov matrix equations and computing values of matrix functions, the two-sided Krylov subspaces have been an active area of research recently, see for instance [5, 73, 74, 77, 79, 109]. In this context, the two-sided Krylov subspaces are usually called *extended* Krylov subspaces.

In [109], Simoncini presents a method for the construction of extended Krylov subspaces which differs from the construction presented in this chapter. In [5] and [77], error estimates for extended Krylov subspaces are given for a wide range of matrix functions. However, in the context of extended Krylov subspaces, the solution of the linear system (2.1), error analysis of the function $f(x) = (1+x)^{-1}$ or the numerical properties of the generated subspaces in finite precision arithmetic have not been previously considered.

2.2 Orthonormal bases for two-sided Krylov subspaces

In what follows, we describe the construction of an orthonormal basis $\{q_1, \ldots, q_{m+2}\}$ for the two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ by using a modified Gram-Schmidt-type process. Originally the construction was presented in [71]. At each step of the resulting two-sided Arnoldi process, we apply both S and S^{-1} once to construct a new pair of basis vectors. It follows that the number of orthonormal basis vectors for the two-sided Krylov subspace increases by two at each step.

We begin by choosing a vector q_1 with $||q_1||_2 = 1$. After this, we compute

$$\hat{q}_2 = Sq_1 - (Sq_1, q_1)q_1, \;\; q_2 = \hat{q}_2/||\hat{q}_2||_2. \tag{2.4}$$

By rearranging the relation (2.4), we then have

$$Sq_1 = \sum_{l=1}^{2} h_{l,1} q_l, \tag{2.5}$$

with $h_{1,1} = (Sq_1, q_1)$ and $h_{2,1} = ||\hat{q}_2||_2$.

Proceeding cyclically we compute, for k odd,

$$\left\{ \begin{array}{rcll} \hat{q}_{k+2} & = & S^{-1}q_k - \sum_{l=1}^{k+1}(S^{-1}q_k, q_l)q_l, & q_{k+2} = \hat{q}_{k+2}/||\hat{q}_{k+2}||_2 \\ \hat{q}_{k+3} & = & Sq_{k+1} - \sum_{l=1}^{k+2}(Sq_{k+1}, q_l)q_l, & q_{k+3} = \hat{q}_{k+3}/||\hat{q}_{k+3}||_2 \end{array} \right. , \tag{2.6}$$

to have a new pair of basis vectors q_{k+2} and q_{k+3}. With these, we now have the orthonormal basis for the two-sided Krylov subspace as $\{q_1, q_2, \ldots, q_{k+2}\}$.

Denote by $t_{l,k}$ and $h_{l,k+1}$ the lth component of the vectors t_k and h_{k+1}, respectively. By rearranging (2.6), we have, for $k = 1, 3, \ldots, m$ odd,

$$\begin{aligned} S^{-1}q_k &= \sum_{l=1}^{k+2} t_{l,k} q_l \\ Sq_{k+1} &= \sum_{l=1}^{k+3} h_{l,k+1} q_l, \end{aligned} \tag{2.7}$$

where t_k and h_{k+1} are both in $\mathbb{C}^{k+3}$ with the entries

$$t_{l,k} = \begin{cases} (S^{-1}q_k, q_l) & , l \leq k+1, \\ ||\hat{q}_{k+2}||_2 & , l = k+2, \end{cases} , \tag{2.8}$$

$$h_{l,k+1} = \begin{cases} (Sq_{k+1}, q_l) & , l \leq k+2, \\ ||\hat{q}_{k+3}||_2 & , l = k+3 \end{cases} , \tag{2.9}$$

and zero otherwise. We note as the iteration proceeds, by the relations (2.8) and (2.9), the length of the previously computed vectors t_l and h_{l+1} with $l < k$ is implicitly increased as new zero elements are added to their end.

Similarly to the standard Arnoldi process, we have a matrix relation in the generated subspace as

$$\begin{aligned} &[Sq_1 \; S^{-1}q_1 \; Sq_2 \; S^{-1}q_3 \; Sq_4 \; \cdots \; S^{-1}q_k \; Sq_{k+1}] \\ &= [q_1 \; q_2 \; q_3 \; q_4 \; \cdots \; q_{k+3}][h_1 \; t_1 \; h_2 \; t_3 \; h_4 \; \cdots \; t_k \; h_{k+1}], \end{aligned} \tag{2.10}$$

where the matrix $Q_{k+3} = [q_1 \; q_2 \; \cdots \; q_{k+3}]$ is of size n-by-$(k+3)$ and has orthonormal columns by construction. The matrix $[h_1 \; t_1 \; h_2 \; t_3 \; \cdots \; t_k \; h_{k+1}]$ is of size $(k+3)$-by-$(k+2)$ and is of an upper Hessenberg form after its last row has been deleted. Notice that the relation (2.10) does not yet yield an Arnoldi-type relation (1.13) of either S or S^{-1} in the basis generated for the two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil \frac{k}{2} \rceil}(S; q_1)$.

To represent the vector Sq_{k+2} in the basis generated for $\mathcal{K}^{\pm}_{\lceil \frac{k}{2} \rceil}(S; q_1)$, it remains to compute $h_{k+2} = Q^*_{k+3} Sq_{k+2}$ for $k = 1, 3, \ldots, m$ odd. We then have the lth component of $h_{l,k+2}$ as

$$h_{l,k+2} \;\; = \;\; (Sq_{k+2}, q_l), \quad l = 1, \ldots, k+3. \tag{2.11}$$

The computational rules (2.4), (2.6) and (2.11) describe how to construct an orthonormal set of basis vectors $\{q_1, q_2, \ldots, q_{k+3}\}$ for $\mathcal{K}^{\pm}_{\lceil \frac{k}{2} \rceil}(S; q_1)$ together with a representation of the matrix S in the basis generated. This construction can also be interpreted as a two-sided Arnoldi process. Using a modified

Algorithm 2.1 Two-sided Arnoldi process

1: Choose a vector q_1 with $||q_1||_2 = 1$
2: $w = Sq_1$, $h_{1,1} = (w, q_1)$, $w = w - h_{1,1}q_1$, $h_{2,1} = ||w||_2$, $q_2 = w/h_{2,1}$
3: **for** $j = 1, 2, \ldots, \frac{m+1}{2}$ **do**
4: $w = S^{-1}q_{2j-1}$
5: **for** $l = 1, \ldots, 2j$ **do**
6: $t_{l,2j-1} = (w, q_l)$, $w = w - t_{l,2j-1}q_l$
7: **end for**
8: $t_{2j+1,2j-1} = ||w||_2$, $q_{2j+1} = w/t_{2j+1,2j-1}$
9: $w = Sq_{2j}$
10: **for** $l = 1, \ldots, 2j+1$ **do**
11: $h_{l,2j} = (w, q_l)$, $w = w - h_{l,2j}q_l$
12: **end for**
13: $h_{2j+2,2j} = ||w||_2$, $q_{2j+2} = w/h_{2j+2,2j}$
14: Compute $w = Sq_{2j+1}$ and $h_{l,2j+1} = (w, q_l)$, for $l = 1, \ldots, 2j+2$
15: **end for**

Gram-Schmidt–type orthogonalization and setting $k = 2j - 1$ for simplicity, we have Algorithm 2.1.

The two-sided Arnoldi process breaks down if $||\hat{q}_{k+2}||_2 = 0$ or $||\hat{q}_{k+3}||_2 = 0$ on lines of 8 and 13 of Algorithm 2.1. Then we have a division by zero and the next basis vector cannot be generated. Assuming that Algorithm 2.1 does not break down, we have the following proposition for the structure of the generated subspace.

Proposition 2.1. *Assume Algorithm 2.1 does not break down before the* $\lceil \frac{m+1}{2} \rceil$*th step has been completed. Then, with* m *odd, the generated set* $\{q_1, \ldots, q_{m+2}\}$ *is an orthonormal basis of*

$$\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1) = \operatorname*{span}_{\mathfrak{p}, \mathfrak{q} \in \mathbb{P}_{\lceil \frac{m}{2} \rceil}} \{\mathfrak{p}(S)q_1, \mathfrak{q}(S^{-1})q_1\}.$$

Proof. The generated set of vectors is orthonormal by construction. It remains to prove that the generated vectors form a basis for the two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$. We have the relation $q_2 = \hat{q}_2/||\hat{q}_2||_q$, where $\hat{q}_2 = Sq_1 - h_{1,1}q_1$ with $h_{1,1} = (Sq_1, q_1)$. In addition, for k odd, we have $q_{k+2} = \hat{q}_{k+2}/||\hat{q}_{k+2}||_2$ and $q_{k+3} = \hat{q}_{k+3}/||\hat{q}_{k+3}||_2$ where $\hat{q}_{k+2}$ and $\hat{q}_{k+3}$ are defined by (2.6).

Denote by $\mathfrak{p}_j$ and $\mathfrak{q}_j$ polynomials of degree j at most. We prove the claim by using induction. For $j = 1$ we have $q_1 = \mathfrak{p}_0(S)q_1$, with $\mathfrak{p}_0(S) = I$.

Similarly, for $q_2 = \hat{q}_2/||\hat{q}_2||_2$ we have,

$$\hat{q}_2 = Sq_1 - h_{1,1}q_1 = S\rho_0(S)q_1 - h_{1,1}\rho_0(S)q_1 = \rho_1(S)q_1,$$

with a polynomial $\rho_1(S)$ of degree one. Using q_1 and q_2, for $q_3 = \hat{q}_3/||\hat{q}_3||_2$ yields

$$\hat{q}_3 = S^{-1}q_1 - t_{1,1}\rho_0(S)q_1 - \frac{t_{2,1}}{||\hat{q}_2||_2}\rho_1(S)q_1 = \tilde{\rho}_1(S)q_1 + \sigma_1(S^{-1})q_1,$$

where $t_{1,1} = (S^{-1}q_1, q_1)$, $t_{2,1} = (S^{-1}q_1, q_2)$ and the polynomials $\tilde{\rho}_1(S)$ and $\sigma_1(S^{-1})$ both of degree one. We then have

$$\text{span}\{q_1, \hat{q}_2, \hat{q}_3\} = \text{span}\{\rho_0(S)q_1, \rho_1(S)q_1, \tilde{\rho}_1(S)q_1 + \sigma_1(S^{-1})q_1\}.$$

Since the dimension of $\text{span}\{\rho_0(S)q_1, \rho_1(S)q_1, \tilde{\rho}_1(S)q_1 + \sigma_1(S^{-1})q_1\}$ equals the dimension of $\mathcal{K}_1^{\pm}(S; q_1)$, it then follows that $\text{span}\{q_1, q_2, q_3\} = \mathcal{K}_1^{\pm}(S; q_1)$ and the claim holds for $j = 1$.

We now assume that the claim holds for two-sided Krylov subspaces $\mathcal{K}^{\pm}_{\lceil \frac{j}{2} \rceil}(S; q_1)$ for all $j \leq k$ odd. By construction of the subspace and the inductive hypothesis, we then have for $j = 1, 3, \ldots, k$ odd

$$\begin{cases} q_{j+1} = \rho_{\lceil \frac{j}{2} \rceil}(S)q_1 + \sigma_{\lceil \frac{j}{2} \rceil - 1}(S^{-1})q_1, \\ q_{j+2} = \rho_{\lceil \frac{j}{2} \rceil}(S)q_1 + \sigma_{\lceil \frac{j}{2} \rceil}(S^{-1})q_1. \end{cases}.$$

Let $t_{l,j}$ and $h_{l,j}$ be defined by the relations (2.9) and (2.11). For $q_{k+3} = \hat{q}_{k+3}/||\hat{q}_{k+3}||_2$ we have

$$\begin{aligned} \hat{q}_{k+3} &= Sq_{k+1} - h_{1,k+1}q_1 - \sum_{l=1}^{k+2} h_{l,k+1}q_l \\ &= S\left(\rho_{\lceil \frac{k}{2} \rceil}(S)q_1 + \sigma_{\lceil \frac{k}{2} \rceil - 1}(S^{-1})q_1\right) - h_{1,k+1}\rho_0(S)q_1 \\ &\quad - \sum_{l=3, l \text{ odd}}^{k+2} h_{l,k+1}\left(\rho_{\lceil \frac{l-2}{2} \rceil}(S)q_1 + \sigma_{\lceil \frac{l-2}{2} \rceil}(S^{-1})q_1\right) \\ &\quad - \sum_{l=2, l \text{ even}}^{k+1} h_{l,k+1}\left(\rho_{\lceil \frac{l-1}{2} \rceil}(S)q_1 + \sigma_{\lceil \frac{l-1}{2} \rceil - 1}(S^{-1})q_1\right), \quad (2.12) \end{aligned}$$

i.e., we can express q_{k+3} as $\mathcal{p}_{\lceil \frac{k}{2} \rceil+1}(S)q_1 + \mathcal{q}_{\lceil \frac{k}{2} \rceil}(S)q_1$. Similarly for $q_{k+4} = \hat{q}_{k+4}/||\hat{q}_{k+4}||_2$, it follows that

$$\begin{aligned}
\hat{q}_{k+4} &= S^{-1}q_{k+2} - t_{1,k+2}q_1 - \sum_{l=1}^{k+3} t_{l,k+2}q_l \qquad (2.13)\\
&= S^{-1}\left(\mathcal{p}_{\lceil \frac{k}{2} \rceil}(S)q_1 + \mathcal{q}_{\lceil \frac{k}{2} \rceil}(S^{-1})q_1\right) - t_{1,k+2}p_0(S)q_1 \\
&\quad - \sum_{l=3,l \text{ odd}}^{k+2} t_{l,k+2}\left(\mathcal{p}_{\lceil \frac{l-2}{2} \rceil}(S)q_1 + \mathcal{q}_{\lceil \frac{l-2}{2} \rceil}(S^{-1})q_1\right) \\
&\quad - \sum_{l=2,l \text{ even}}^{k+3} t_{l,k+2}\left(\mathcal{p}_{\lceil \frac{l-1}{2} \rceil}(S)q_1 + \mathcal{q}_{\lceil \frac{l-1}{2} \rceil-1}(S^{-1})q_1\right), \qquad (2.14)
\end{aligned}$$

i.e., we can express q_{k+4} as $\mathcal{p}_{\lceil \frac{k}{2} \rceil+1}(S)q_1 + \mathcal{q}_{\lceil \frac{k}{2} \rceil+1}(S)q_1$. It then follows that

$$\text{span}\{q_1, \ldots, q_{k+2}, q_{k+3}, q_{k+4}\} = \underset{\mathcal{p},\mathcal{q}\in\mathbb{P}_{\lceil \frac{k}{2} \rceil+1}}{\text{span}} \{\mathcal{p}(S)q_1, \mathcal{q}(S)q_1\},$$

where $\mathbb{P}_j$ denotes the set of polynomials of degree j at most. Again, since the dimension of the set $\text{span}_{\mathcal{p},\mathcal{q}\in\mathbb{P}_{\lceil \frac{k}{2} \rceil+1}}\{\mathcal{p}(S)q_1, \mathcal{q}(S)q_1\}$ equals the dimension of $\mathcal{K}^{\pm}_{\lceil \frac{k}{2} \rceil+1}(S; q_1)$, the span of the set $\{q_1, \ldots, q_{k+2}, q_{k+3}, q_{k+4}\}$ is equal to the span of vectors in the two-sided Krylov subspace $\mathcal{K}_{\lceil \frac{k}{2} \rceil+1}(S; q_1)$ and the induction is complete. □

As a consequence of the proof of Proposition 2.1, we have the following corollary for the structure of the generated basis vectors.

Corollary 2.2. *Let $k = 1, 3, 5, \ldots, m$, with m odd. By the construction of the two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$, the vectors q_{k+1} and q_{k+2} can be expressed as*

$$\begin{cases} q_{k+1} = \mathcal{p}_{\lceil \frac{k}{2} \rceil}(S)q_1 + \mathcal{q}_{\lceil \frac{k}{2} \rceil-1}(S^{-1})q_1, \\ q_{k+2} = \mathcal{p}_{\lceil \frac{k}{2} \rceil}(S)q_1 + \mathcal{q}_{\lceil \frac{k}{2} \rceil}(S^{-1})q_1, \end{cases}$$

where p_j and q_j denote polynomials of degree j at most.

Algorithm 2.1 constructs a relation similar to the standard Arnoldi's method as follows.

Proposition 2.3. *Let $Q_{m+3} = [q_1\ q_2\ \cdots\ q_{m+3}]$ be the matrix having the orthonormal basis vectors q_j of the two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ as its columns. The two-sided Arnoldi process, i.e., Algorithm 2.1, produces the relation*

$$SQ_{m+2} = Q_{m+3}\bar{H}_{m+2}, \tag{2.15}$$

where $\bar{H}_{m+2}$ is of size $(m+3)$-by-$(m+2)$.

Denote by H_{m+2} the square matrix obtained after the last row of $\bar{H}_{m+2} = [h_1\ h_2\ h_3\ h_4\ \cdots\ h_{m+2}]$ has been deleted. Then

$$H_{m+2} = \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} & h_{1,4} & \cdots & h_{1,m} & h_{1,m+1} & h_{1,m+2} \\ h_{2,1} & h_{2,2} & h_{2,3} & h_{2,4} & \cdots & h_{2,m} & h_{2,m+1} & h_{2,m+2} \\ 0 & h_{3,2} & h_{3,3} & h_{3,4} & \cdots & h_{3,m} & h_{3,m+1} & h_{3,m+2} \\ 0 & h_{4,2} & h_{4,3} & h_{4,4} & \cdots & h_{4,m} & h_{4,m+1} & h_{4,m+2} \\ 0 & 0 & 0 & h_{5,4} & \cdots & h_{5,m} & h_{5,m+1} & h_{5,m+2} \\ 0 & 0 & 0 & h_{6,4} & \cdots & h_{6,m} & h_{6,m+1} & h_{6,m+2} \\ 0 & 0 & 0 & 0 & \ddots & h_{7,m} & h_{7,m+1} & h_{7,m+2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & h_{m+1,m-1} & h_{m+1,m} & h_{m+1,m+1} & h_{m+1,m+2} \\ 0 & 0 & 0 & 0 & 0 & 0 & h_{m+2,m+1} & h_{m+2,m+2} \end{bmatrix},$$

is a Hessenberg matrix except that the entries $(2j+2, 2j)$ are nonzero in general for $j \geq 1$.

Proof. By writing the relation (2.5) in a matrix form we have $Sq_1 = Q_2 h_1$. Similarly, from (2.7) we have the matrix form of Sq_{k+1} as

$$Sq_{k+1} = Q_{k+3}h_{k+1}. \tag{2.16}$$

On the other hand, since Q_{k+3} has orthonormal columns, a matrix form of (2.11) directly yields

$$Sq_{k+2} = Q_{k+3}h_{k+2}.$$

Combining expressions for Sq_1, Sq_{k+1} and Sq_{k+2} for $k = 1, \ldots, m$, we have the relation

$$[Sq_1\ Sq_2\ \cdots\ Sq_{m+1}\ Sq_{m+2}] = [Q_2h_1\ Q_4h_2\ \cdots\ Q_{m+3}h_{m+1}\ Q_{m+3}h_{m+2}].$$

When written in a matrix form this yields $SQ_{m+2} = Q_{m+3}\bar{H}_{m+2}$, with $\bar{H}_{m+2} = [h_1\ h_2\ \cdots\ h_{m+2}]$. The square matrix H_{m+2} obtained after deleting the last row of $\bar{H}_{m+2}$ has a Hessenberg structure with the entries $(2j+2, 2j)$ nonzero for $j \geq 1$. □

If either one of the basis vectors q_{k+2} or q_{k+3} on lines 8 or 13 of Algorithm 2.1 cannot be generated, the two-sided Arnoldi process breaks down and we have generated a minimal polynomial of S at q_1. Recall that the minimal polynomial of S at q_1 is the monic polynomial $\wp$ of the least degree satisfying $\wp(S)q_1 = 0$. We have the following proposition.

Proposition 2.4. *The two-sided Arnoldi process, i.e., Algorithm 2.1 breaks down at step k if and only if the minimal polynomial of the matrix S at q_1 is of degree $k+1$ or $k+2$.*

Proof. The breakdown occurs if $||\hat{q}_{k+2}||_2 = 0$ (or $||\hat{q}_{k+3}||_2 = 0$). Then the next basis vector cannot be generated. For $k+2$ this implies, by (2.6), the relation

$$0 = \hat{q}_{k+2} = S^{-1}q_k - \sum_{l=1}^{k+1}(S^{-1}q_k, q_l)q_l. \tag{2.17}$$

By the Proposition 2.1 and the Corollary 2.2, we can write q_j for $j = 1, \ldots, k+1$ in the basis generated for $\mathcal{K}^{\pm}_{\lceil \frac{k}{2} \rceil}(S; q_1)$ as $q_{j+1} = \wp_{\lceil \frac{j}{2} \rceil}(S)q_1 + \mathfrak{q}_{\lceil \frac{j}{2} \rceil - 1}(S^{-1})q_1$ and $q_{j+2} = \wp_{\lceil \frac{j}{2} \rceil}(S)q_1 + \mathfrak{q}_{\lceil \frac{j}{2} \rceil}(S^{-1})q_1$, where $\wp_j(S)$ and $\mathfrak{q}_j(S^{-1})$ denote polynomials of $\wp_j$ and $\mathfrak{q}_j$ in S and S^{-1} of degree j at most. From (2.17), we then have

$$\wp_{\lceil \frac{k}{2} \rceil}(S)q_1 + \mathfrak{q}_{\lceil \frac{k}{2} \rceil}(S^{-1})q_1 = 0.$$

Multiplication by $S^{\lceil \frac{k}{2} \rceil}$ from the left yields

$$S^{\lceil \frac{k}{2} \rceil}\left(\wp_{\lceil \frac{k}{2} \rceil}(S)q_1 + \mathfrak{q}_{\lceil \frac{k}{2} \rceil}(S^{-1})q_1\right) = \wp_{k+1}(S)q_1 = 0,$$

where $\wp_{k+1}(S)$ denotes a polynomial in S of degree $k+1$. Thus it follows that if $||\hat{q}_{k+2}||_2 = 0$, the minimal polynomial of S at q_1 is of degree $k+1$, since having a minimal polynomial of S at q_1 with a degree greater than $k+1$ would imply $||\hat{q}_{k+2}||_2 \neq 0$, which is a contradiction.

For the necessary part, consider the case $\wp_{k+1}(S)q_1 = 0$. The degree of the polynomial cannot be any smaller that $k+1$, since by the first part of this proof the construction would have stopped at an earlier step. After multiplication of $\wp_{k+1}(S)q_1$ from the left by $S^{-\lceil \frac{k}{2} \rceil}$, we have

$$\wp_{\lceil \frac{k}{2} \rceil}(S)q_1 + \mathfrak{q}_{\lceil \frac{k}{2} \rceil}(S^{-1})q_1 = 0.$$

By Corollary 2.2 it then follows that $||\hat{q}_{k+2}||_2 = 0$.

The proof for the case $||\hat{q}_{k+3}||_2 = 0$ having the minimal polynomial of degree $k+2$ is similar.

□

In exact arithmetic, to have a representation of Sq_{k+2} in the basis generated for $k = 1, \ldots, m$ odd, it is sufficient to compute inner products with only the first $k+3$ basis vectors. This motivates the following lemma.

Lemma 2.5. *In exact arithmetic the vectors Sq_{k+2}, for $k = 1, 3, \ldots, m$, can be represented in the set of basis vectors $\{q_1, \ldots, q_{k+3}\}$.*

Proof. By the first part of the relation (2.7), after multiplying by S from the left we have

$$Sq_{k+2} = \frac{1}{||\hat{q}_{k+2}||_2} \left(q_k - \sum_{l=1}^{k+1} (S^{-1}q_k, q_l) Sq_l \right). \tag{2.18}$$

By Proposition 2.3, the products Sq_{k+1} can be represented with up to $k+3$ coefficients in the basis generated for $k = 1, 3, \ldots, m$. We now show the claim by induction. For $k = 1$, we have

$$Sq_3 = \frac{1}{||\hat{q}_3||_2} \left(q_1 - \sum_{l=1}^{2} (S^{-1}q_1, q_l) Sq_l \right).$$

Since the vector Sq_2 can be represented with up to 4 coefficients in the basis $\{q_1, q_2, q_3, q_4\}$, the vector Sq_4 can be represented with 4 coefficients at most in the same basis.

Now assume that the claim hold up to $k = j-2 \geq 1$, i.e., we can represent the products Sq_j in the basis $\{q_1, \ldots, q_{j+1}\}$ with up to $j+1$ coefficients. By the construction of q_{j+3} and the latter part of the relation (2.7), for Sq_{j+1} then have a representation in the basis $\{q_1, \ldots, q_{j+3}\}$ with up to $j+3$ nonzero coefficients. Thus, for the generated basis vector q_{j+2}, the relation (2.18) states that the product Sq_{j+2} can be represented with $j+3$ nonzero coefficients at most in the basis $\{q_1, \ldots, q_{j+3}\}$ and the induction is complete.

□

We note that Lemma 2.5 enables a more efficient way of computing h_{k+2}. This is presented in more detail in Chapter 4.

In this section, we have described the two-sided Arnoldi method for the construction of the two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$. In the following section, we consider approximating the solution of the linear system (2.1) by using two-sided Krylov subspaces.

2.3 Derivation of TSMRES

Denote by $r_0 = b-(I+S)x_0$ the initial residual with an initial approximation x_0 to the solution of the linear system (2.1). We first set $q_1 = r_0/||r_0||_2$ and then construct an orthonormal basis for the two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ with the two-sided Arnoldi process. After completing $\lceil \frac{m}{2} \rceil$ steps, we have a matrix relation of the form (2.15).

To approximately solve the linear system (2.1), for a GMRES -type of approach, we must construct a representation for the matrix $(I + S)$ in the two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$. By using the Proposition 2.3 and the relation (2.15), it follows that we have a representation of $(I + S)$ in the basis generated for $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ as

$$(I + S)Q_{m+2} = Q_{m+3}(\bar{I} + \bar{H}_{m+2}), \tag{2.19}$$

with

$$\bar{I} = \begin{bmatrix} I_{m+2} \\ 0 \cdots 0 \end{bmatrix},$$

where I_{m+2} denotes an $(m + 2)$-by-$(m + 2)$ identity matrix.

With the matrix representation (2.19), we now consider solving the minimization problem (2.3) in the two-sided Krylov subspace. By again following Huhtanen and Nevanlinna [71], for a minimal residual approach, we have

$$\begin{aligned} &\min_{v \in \mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S;q_1)} ||b - (I + S)(x_0 + v)||_2 \\ &= \min_{y_{m+2} \in \mathbb{C}^{m+2}} ||b - (I + S)(x_0 + Q_{m+2}y_{m+2})||_2 \\ &= \min_{y_{m+2} \in \mathbb{C}^{m+2}} ||\beta e_1 - (\bar{I} + \bar{H}_{m+2})y_{m+2}||_2, \end{aligned} \tag{2.20}$$

where $\bar{H}_{m+2}$ and $\bar{I}$ as before and $\beta = ||r_0||_2$. To find the minimizer y_{m+2}, we need to solve a $(m + 3)$-by-$(m + 2)$ least-squares problem. Since generally $m \ll n$, finding y_{m+2} is computationally inexpensive and (2.20) can be solved in $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ without a significant effort. Once the minimizer has been computed, we have the approximate solution as $x_m = x_0 + z_m = x_0 + Q_{m+2}y_{m+2}$.

Again to simplify our notation, we denote $k = 2j - 1$. By using the Algorithm 2.1 and equations (2.19) and (2.20), we obtain a two-sided minimal residual algorithm, or simply TSMRES as presented in Algorithm 2.2.

Efficient implementation and computational complexity of the TSMRES method are discussed in more detail in Section 4.3. We also briefly consider, in Section 4.5, the numerical stability of TSMRES.

Algorithm 2.2 TSMRES

Compute $r_0 = b - x_0 - Sx_0$, $\beta = ||r_0||_2$, $q_1 = r_0/\beta$
$w = Sq_1$, $h_{1,1} = (w, q_1)$, $w = w - h_{1,1}q_1$, $h_{2,1} = ||w||_2$, $q_2 = w/h_{2,1}$
for $j = 1, 2, \ldots, \frac{m+1}{2}$ **do**
    $w = S^{-1}q_{2j-1}$
    for $l = 1, \ldots, 2j$ **do**
        $t_{l,2j-1} = (w, q_l)$, $w = w - t_{l,2j-1}q_l$
    end for
    $t_{2j+1,2j-1} = ||w||_2$, $q_{2j+1} = w/t_{2j+1,2j-1}$
    $w = Sq_{2j}$
    for $l = 1, \ldots, 2j+1$ **do**
        $h_{l,2j} = (w, q_l)$, $w = w - h_{l,2j}q_l$
    end for
    $h_{2j+2,2j} = ||w||_2$, $q_{2j+2} = w/h_{2j+2,2j}$
    Compute $w = Sq_{2j+1}$ and $h_{l,2j+1} = (w, q_l)$, for $l = 1, \ldots, 2j+2$
end for
Compute $y_{2m+1} = \text{Argmin}_y||\beta e_1 - (\bar{I} + \bar{H}_{2m+1})y||_2$
$x_m = x_0 + Q_{2m+1}y_{2m+1}$

In the next sections we discuss the properties and the speed of convergence of TSMRES. We are particularly interested in the effect of using a two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ instead of the standard Krylov subspace $\mathcal{K}_{m+2}(S; q_1)$.

2.4 Algebraic properties of TSMRES

The two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ used by TSMRES is different from an equal sized standard Krylov-subspace $\mathcal{K}_{m+2}(S; q_1)$ used by GMRES. Since TSMRES minimizes the residual norm in $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$, it is optimal but different from GMRES. In this section, we analyze some algebraic properties of the two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ and the TSMRES method.

By Propositions 2.1 and 2.3, the two-sided Arnoldi process, i.e., Algorithm 2.1 generates an orthonormal basis for the two-sided Krylov subspace. In addition, it computes a matrix representation of S in the basis generated.

Consider a monic polynomial of degree k, defined as

$$\pitchfork_k(x) = x^k + c_{k-1}x^{k-1} + \cdots + c_1 x + c_0 = \sum_{j=0}^{k} c_j x^j,$$

with coefficients c_j, for $j = 1, \ldots, k$ with $c_k = 1$. Let j be the index of the term with nonzero coefficient c_j of the smallest degree x^j. We then have, after multiplication by $c_j^{-1}S^{-k}$ from the left

$$c_j^{-1}S^{-k}\pi_k(S) = \tilde{c}_0 + \tilde{c}_1 S^{-1} + \cdots + \tilde{c}_{k+(j-1)}S^{-k+(j+1)} + S^{-k+j} = \phi_{k-j}(S^{-1}),$$

where ϕ_{k-j} denotes a monic polynomial of degree $k - j$.

In the previous section we have assumed that we could generate the two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ by using Algorithm 2.1 up to some predefined m. However, if $||\hat{q}_{k+2}||_2 = 0$ or $||\hat{q}_{k+3}||_2 = 0$, we have a division by zero and the next basis vector cannot be generated. In TSMRES this breakdown happens on lines 8 or 13. For this, we have the following proposition.

Proposition 2.6. *Let $(I + S)$ be nonsingular.* TSMRES *breaks down at step k odd if and only if the approximate solution x_{k+1} (or x_{k+2}) is exact, i.e., $||b - x_{k+1} - Sx_{k+1}||_2 = 0$.*

Proof. When $||\hat{q}_{k+2}||_2 = 0$, Proposition 2.4 implies that the degree of the minimal polynomial of q_1 is $k + 1$. Then we have

$$\pi_{k+1}(S)q_1 = 0, \tag{2.21}$$

where $\pi_{k+1}(S)$ denotes a polynomial of S of degree $k + 1$. We now show that the relation (2.21) implies the exactness of the solution x_k. Observe that the degree of polynomial $\pi_{k+1}(S)q_1$ is translation invariant. Thus we have $\tilde{\pi}_{k+1}(I + S)q_1 = 0$, with $\tilde{\pi}_{k+1}(I + S)q_1$ having same degree but different coefficients than $\pi_{k+1}(S)q_1$. It then follows that

$$\tilde{\pi}_{k+1}(I + S)q_1 = 0 \Longleftrightarrow (I + S)^{-1}q_1 = \tilde{\pi}_k(I + S)q_1,$$

where $\tilde{\pi}_k(I + S)$ is of degree k. Note that having $(I + S)^{-1}q_1 = \tilde{\pi}_j(I + S)q_1$ with $j < k$ would contradict the minimality of $\pi_{k+1}(S)q_1$. Since $q_1 = r_0/||r_0||_2$ with $r_0 = b - (I + S)x_0$, we have

$$(I + S)^{-1}b = x_0 + \tilde{\pi}_k(I + S)r_0,$$

by which we set $z_k = \tilde{\pi}_k(I + S)r_0$. Thus, a breakdown in the construction of q_{k+2} implies the exactness of the approximate solution $x_k = x_0 + z_k$.

To show the necessary condition, we start from $r_k = b - (I + S)(x_0 + y)$ and observe that $||r_k||_2 = 0$ implies

$$(I + S)y = r_0 \Longleftrightarrow (I + S)^{-1}r_0 = \tilde{\pi}_k(I + S)r_0,$$

from which we have $\tilde{\pitchfork}_{k+1}(I+S)q_1 = 0$. By the first part of this proof, the degree of the minimal polynomial must be equal to $k+1$, since otherwise construction would have stopped earlier. Since the degree of a polynomial is translation invariant, we again have that the minimal polynomial of q_1 is of the form $\pitchfork_{k+1}(S)q_1 = 0$. By Proposition 2.4, this implies a breakdown at step k with $||\hat{q}_{k+2}||_2 = 0$ in the construction of the two-sided Krylov subspace.

The proof for the case $||\hat{q}_{k+3}||_2 = 0$ is similar. □

As a numerical example of the breakdown of TSMRES, let us look at the following.

Example 2.1 We take a set of n linearly independent normally distributed random vectors $T = [t_1, t_2, \ldots, t_n]$ of length n generated with MATLAB. We then form the QR factorization of T as $T = QR$ to obtain an orthonormal set of basis vectors $Q = [q_1, q_2, \ldots, q_n]$. We construct normally distributed random $\lambda_j \in \mathbb{C}$, $j = 1, \ldots, k$ and define a diagonal n-times-n matrix D as

$$D = \begin{bmatrix} \lambda_1 & & & & & & \\ & \ddots & & & & & \\ & & \lambda_k & & & & \\ & & & 1 & & & \\ & & & & \ddots & & \\ & & & & & 1 & \end{bmatrix}.$$

With these, we set $S = QDQ^*$. Since Q is unitary we have $S^{-1} = QD^{-1}Q^*$ with D^{-1} diagonal. In addition, $I+S$ has only $k+1$ distinct eigenvalues and its minimal polynomial is of degree $k+2$.

We set $n = 100$ and $k = 21$ and $b = \hat{b}/||\hat{b}||_2$ where $\hat{b}$ is a vector of all ones of size n and S defined as above. We then solve the linear system (2.1) with both GMRES and TSMRES. After each step, we compute the norms of the residual $||b - (I+S)x_j||_2$ for both algorithms. Figure 2.1 presents the computed relative residuals after computing subspaces $\mathcal{K}_j(I+S; b)$ for GMRES and $\mathcal{K}^{\pm}_{\lceil \frac{j}{2} \rceil}(S; b)$ for TSMRES.

From Figure 2.1 we observe that both GMRES and TSMRES break down after the size of the subspace has reached 23, i.e., $k+2$. By the construction of S, this behaviour is to be expected for GMRES [105, Proposition 6.10, pp. 171] and for TSMRES by Proposition 2.6. Note that due to finite precision arithmetics, for both algorithms the final residual is very small, but nonzero.

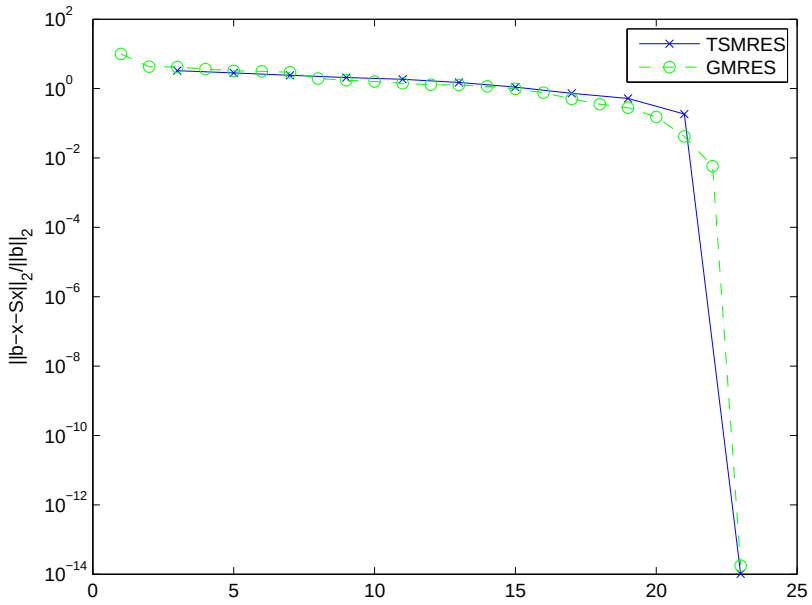


Figure 2.1: Norm of the residual for GMRES (dashed green line) and TSMRES (solid blue line)

In order to be able to apply the TSMRES method to a linear system of the form (2.1), the matrix S needs to be readily invertible. We deem a matrix as being readily invertible if solving linear systems with it is inexpensive in terms of floating point operations. By this we mean that solving a linear system involving a matrix is allowed to be approximately $O(\log(n))$ times more expensive than multiplying a vector with the same matrix. In the case of sparse matrices, an acceptable computational complexity of solving a linear system matrices involving readily invertible matrices is usually $O(n\log(n))$ at most. Otherwise, the cost of forming two-sided Krylov subspaces quickly becomes prohibitive.

In the following, we give some examples of linear systems where the structure of the form (2.1) is either directly present or becomes available after preconditioning the system. In the latter case, we consider linear system $Ax = c$ and splitting the matrix as $A = L + R$, where both parts are nonsingular and readily invertible. After preconditioning by L^{-1} from the left, we have (2.1) with $b = L^{-1}c$ and $S = L^{-1}R$ readily invertible.

Example 2.2 The most generic type of splittings are purely algebraic Gauss-Seidel–type of splittings of the form

$$A = \hat{L} + D + \hat{R}, \tag{2.22}$$

where D is a block diagonal matrix and $\hat{L}$ and $\hat{R}$ are strictly lower and upper triangular block matrices. The freedom is to split the elements in the diagonal blocks of D between the two matrices, i.e.,

$$L = \alpha D + \hat{L}, \quad R = (1-\alpha)D + \hat{R}, \tag{2.23}$$

with some parameter $\alpha \in \mathbb{R}$. After preconditioning by the matrix L, we have S which is readily invertible in approximately $O(\mathrm{nz}(A))$ operations, where $\mathrm{nz}(A)$ denotes the number of nonzero elements in A.

Another type of purely algebraic splittings of the form (2.22) are k-Hessenberg splittings. Denote by $a_{i,j}$ an entry at position (i,j) in the matrix A. We define L_k as the strictly lower k-diagonal part of A having the entries

$$l_{i,j} = \begin{cases} a_{i,j} & ,(i-k) > j \\ 0 & , \text{ otherwise} \end{cases} \tag{2.24}$$

and similarly R_k as the strictly upper k-diagonal part of A having the entries

$$r_{i,j} = \begin{cases} a_{i,j} & ,(j-k) > i \\ 0 & , \text{ otherwise} \end{cases}, \tag{2.25}$$

i.e., L_k and R_k contain those entries of A which are below and above the kth lower and upper diagonals of A, respectively. Let D be the band matrix with upper and lower bandwidths k having the entries of A in this band and $\hat{L} = L_k$ and $\hat{R} = R_k$. With these, by using the relation (2.23), we have a splitting where L and R are lower and upper k-Hessenberg matrices, respectively.

In Chapter 6 we consider a direct method for the solution of sparse k-Hessenberg linear systems in $O((k+1)\mathrm{nz}(A))$ floating point operations, where $\mathrm{nz}(A)$ denotes the number of nonzero elements in A as before. Therefore, after preconditioning by the matrix L, we have S which is readily invertible in $O((k+1)\mathrm{nz}(A))$ operations.

Example 2.3 Scattering problems in two and three dimensions for the Helmholz equation in the inhomogeneous media are equivalent to the Lippmann-Schwinger integral equations [118]. In a matrix form, the discretized equations can then be written as

$$(I+S)x = b, \quad S = D_a \mathcal{F}_n^{-1} \hat{K}_n \mathcal{F}_n,$$

where D_a and $\hat{K}_n$ are diagonal matrices and $\mathcal{F}_n$ is the Fourier basis matrix. Using fast Fourier transform, S is readily invertible in $O(n\log(n))$ operations.

Example 2.4 In ADI-type of iterations for two dimensional elliptic equations with separable coefficients, a discretization in a grid is split into x- and y-directions as

$$(H+V)x = c$$

where matrix H is block diagonal matrix with tridiagonal blocks and matrix V is block tridiagonal with diagonal blocks (or vice versa if the ordering on the grid is reversed) [105, 111, 112]. Then, after preconditioning from the left by H, we have S which is readily invertible in $O(n)$ operations. We note that ADI methods seem to have an interesting connection to the solution of Lyapunov matrix equations, see [41, 127].

Example 2.5 Linear systems of the form $(\lambda I + M)x = b$ with a shift parameter $\lambda \in \mathbb{C}$ often arise in eigenvalue computations. For a symplectic matrix M of size $2n$-by-$2n$, we have

$$MJM^T = J$$

with

$$J = \begin{pmatrix} 0 & I_n \\ -I_n & 0 \end{pmatrix},$$

where I_n denotes an n-by-n identity matrix. For the inverse of J it holds $J^{-1} = J^T = -J$, so $M^{-1} = -JM^TJ$ and therefore M is readily invertible. Symplectic matrices arise typically in Hamiltonian mechanics. In numerical computations involving symplectic operators, it is advisable to use symplectic numerical methods to numerically preserve physical properties such as Hamiltonian flow on the manifold [44].

For other examples of splittings with readily invertible parts we refer to [71].

2.5 Theoretical convergence of TSMRES

In this section, we consider the convergence properties of TSMRES. In addition to studying convergence theoretically, we conduct numerical experiments to compare the two-sided Krylov subspaces $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ with the standard Krylov subspaces $\mathcal{K}_{m+2}(S; q_1)$ used by GMRES.

We begin by deriving a standard upper bound on the norm of the residual of TSMRES.

Proposition 2.7. *Let S be diagonalizable with $S = XDX^{-1}$. For the residual norm of* TSMRES *we then have*

$$\begin{aligned} &\min_{v \in \mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S;q_1)} ||b - (I+S)v||_2 \leq \\ &\min_{\mathcal{p},\mathcal{q} \in \mathbb{P}_{\lceil \frac{m}{2} \rceil}} \max_{\lambda \in \sigma(S)} |1 - (1+\lambda)(\mathcal{p}(\lambda) + \mathcal{q}(\lambda^{-1}))|\ \kappa(X)||r_0||_2, \end{aligned} \tag{2.26}$$

where $\mathbb{P}_j$ denotes the set of polynomials of degree j at most, $\sigma(S)$ the spectrum of S and $\kappa(X) = ||X||_2||X^{-1}||_2$ the condition number of X.

Proof. We verify the claim by a direct calculation. It follows that

$$\begin{aligned} &\min_{v \in \mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S;r_0)} ||b - (I+S)v||_2 \\ &= \min_{\mathcal{p},\mathcal{q} \in \mathbb{P}_{\lceil \frac{m}{2} \rceil}} ||b - (I+S)(\mathcal{p}(S) + \mathcal{q}(S^{-1}))b||_2 \\ &= \min_{\mathcal{p},\mathcal{q} \in \mathbb{P}_{\lceil \frac{m}{2} \rceil}} ||X(I - (I+D)(\mathcal{p}(D) + \mathcal{q}(D^{-1})))X^{-1}b||_2 \\ &\leq \min_{\mathcal{p},\mathcal{q} \in \mathbb{P}_{\lceil \frac{m}{2} \rceil}} ||X||_2||I - (I+D)(\mathcal{p}(D) + \mathcal{q}(D^{-1}))||_2||X^{-1}||_2||b||_2 \\ &= \min_{\mathcal{p},\mathcal{q} \in \mathbb{P}_{\lceil \frac{m}{2} \rceil}} ||I - (I+D)(\mathcal{p}(D) + \mathcal{q}(D^{-1}))||_2\kappa(X)||b||_2. \end{aligned} \tag{2.27}$$

Since $I - (I+D)(\mathcal{p}(D) + \mathcal{q}(D^{-1}))$ is diagonal, by the definition of the two-norm we have

$$||I - (I+D)(\mathcal{p}(D) + \mathcal{q}(D^{-1}))||_2 = \max_{\lambda \in \sigma(S)} |1 - (1+\lambda)(\mathcal{p}(\lambda) + \mathcal{q}(\lambda^{-1}))|,$$

and the claim follows. □

We note that by the third line of the proof of Proposition 2.7, with the choice $\mathcal{q} \equiv 0$ we have the standard GMRES residual norm for a Krylov subspace of size $2m+1$ and the following corollary follows.

Corollary 2.8. *For the residual vectors r^{ts}_{2m} of* TSMRES *and r^{gm}_{m} of* GMRES *it holds*

$$||r^{ts}_{2m}||_2 \leq ||r^{gm}_{m}||_2, \tag{2.28}$$

i.e, the residual norm of TSMRES *at the step $2m$ is at least as good as that of* GMRES *at the step m.*

The Proposition 2.7 and the Corollary 2.8 relate the residual of TSMRES to the residual of standard GMRES. However, since S is readily invertible, we can equally well consider solving the right-preconditioned linear system

$$(I + S^{-1})y = b, \quad x = S^{-1}y.$$

Then we have the residual as $r = b - (I + S^{-1})y = b - (I + S^{-1})Sx = b - (I + S)x$, which is identical to the original unpreconditioned residual. Similarly to (1.16), at the mth step of iteration, a correction z_m for the iterate $y_m = y_0 + z_m$ is determined in the Krylov subspace $\mathcal{K}_m(S^{-1}; q_1)$ by solving a minimization problem

$$\min_{v \in \mathcal{K}_m(S^{-1}; r_0)} ||b - (I + S^{-1})(y_0 + v)||_2,$$

with $q_1 = r_0/||r_0||_2$ and the initial residual $r_0 = b - (I + S^{-1})y_0$. Multiplying the computed approximate solution y_m by S^{-1} from the left yields

$$S^{-1}y_m = S^{-1}y_0 + S^{-1}z_m \iff x_m = x_0 + S^{-1}z_m.$$

after using $x_m = S^{-1}y_m$. We denote this right-preconditioned GMRES as GMRES$-S^{-1}$, since it generates the subspace $\mathcal{K}_m(S^{-1}; q_1)$.

We now observe that after setting $\curvearrowright \equiv 0$ in the third line of the proof of Proposition 2.7, we have GMRES$-S^{-1}$ residual for subspace of size $2m + 1$.

Corollary 2.9. *For the residual vectors* r_{2m}^{ts} *of* TSMRES *and* $r_m^{gm-S^{-1}}$ *of* GMRES$-S^{-1}$*, it holds*

$$||r_{2m}^{ts}||_2 \leq ||r_m^{gm-S^{-1}}||_2, \tag{2.29}$$

i.e, the residual norm of TSMRES *at the step* $2m$ *is at least as good as that of* GMRES$-S^{-1}$ *at the step* m*.*

Recall that in the case S is not diagonalizable, or when the degree of nonnormality is not mild, i.e., $\kappa(X)$ is large, for GMRES the relation (1.17) states that the eigenvalues may not provide a good estimate of the rate of convergence. Several alternatives to assess the speed of convergence have been proposed [33]. In the general case, however, the spectrum may not give any useful information at all, see [56].

For TSMRES similar conclusion apply. From Proposition 2.7 we observe that in the case $\kappa(X)$ is not small, bound (2.26) is of little value in practice. With TSMRES, we do not pursue the other approaches any further.

2.6 Numerical illustrations of the polynomial approximation problem

In studying the convergence properties of the two-sided Krylov subspaces, understanding the approximation problem

$$\min_{\mathcal{p},\mathcal{q}\in\mathbb{P}_{\lceil \frac{m}{2} \rceil}} \max_{\lambda\in\sigma(S)} |1-(1+\lambda)(\mathcal{p}(\lambda)+\mathcal{q}(\lambda^{-1}))| \tag{2.30}$$

is of importance. It is challenging to construct direct theoretical estimates to the approximation problem (2.30). For any $\mathcal{p}$ and $\mathcal{q}$, we can write

$$\begin{aligned} &\max_{\lambda\in\sigma(S)} |1-(1+\lambda)(\mathcal{p}(\lambda)+\mathcal{q}(\lambda^{-1}))| \\ &= \max_{\lambda\in\sigma(S)} |(1+\lambda)(\frac{1}{1+\lambda}-(\mathcal{p}(\lambda)+\mathcal{q}(\lambda^{-1})))| \\ &\leq \max_{\lambda\in\sigma(S)} |(\frac{1}{1+\lambda}-(\mathcal{p}(\lambda)+\mathcal{q}(\lambda^{-1})))||(1+\lambda)| \end{aligned} \tag{2.31}$$

Set $M = \max_{\lambda\in\sigma(S)} |(1+\lambda)|$. Then

$$\begin{aligned} &\min_{\mathcal{p},\mathcal{q}\in\mathbb{P}_{\lceil \frac{m}{2} \rceil}} \max_{\lambda\in\sigma(S)} |1-(1+\lambda)(\mathcal{p}(\lambda)+\mathcal{q}(\lambda^{-1}))| \\ &\leq M \min_{\mathcal{p},\mathcal{q}\in\mathbb{P}_{\lceil \frac{m}{2} \rceil}} \max_{\lambda\in\sigma(S)} |(\frac{1}{1+\lambda}-(\mathcal{p}(\lambda)+\mathcal{q}(\lambda^{-1})))|. \end{aligned} \tag{2.32}$$

Denote by $\mathcal{S}^2$ the Riemann Sphere, i.e., the set containing $\mathbb{R}^2$ and $\{\infty\}$ and by $f \in H(U)$ the class of all holomorphic functions f in U. We now recall the Runge's theorem, stated as in Rudin [98, pp. 270].

Theorem 2.10. *Suppose K is a compact set in the plane and $\{\alpha_j\}$ is a set which contains one point in each component of $\mathcal{S}^2 \setminus K$. If Ω is open, $\Omega \supset K$, $f \in H(\Omega)$, and $\epsilon > 0$, there exists a rational function R, all of whose poles lie in the prescribed set $\{\alpha_j\}$, such that*

$$|f(\lambda) - R(\lambda)| \leq \epsilon \tag{2.33}$$

for every $\lambda \in K$

We now assume that the discrete spectrum $\sigma(S)$ is replaced by a compact set K including $\sigma(S)$. Since S is assumed to be invertible, we require $0 \notin K$. Then the bound (2.32) used with Runge's theorem gives some insight into the convergence properties of TSMRES.

Let $f(\lambda) = 1/(1+\lambda)$. When the points -1 and 0 belong to the same component of $\mathcal{S}^2 \setminus K$, we can set α_1 as 0 and α_2 as ∞ to have a point in the components of $\mathcal{S}^2 \setminus K$ where there is a pole of f. Then, by Runge's theorem approximation by a rational function is feasible.

Similarly, when -1 and ∞ belong to the same component of $\mathcal{S}^2 \setminus K$, we can approximate f with polynomials. Otherwise Runge's theorem cannot be applied and the approximation by a rational function $p(z) + q(z^{-1})$ cannot be guaranteed.

These theoretical convergence results are concerned only with the case of continuum. In practice, $\mathcal{S}^2 \setminus K$ contains only one component. In the following, we attempt to gain insight in the convergence properties of TSMRES by numerical experiments.

We choose S to be a diagonal matrix and set its eigenvalues to match the case of interest. Then, by the equation (2.27), for a similar matrix $\hat{S} = XSX^{-1}$ the convergence behaviour is similar when the condition number of the similarity transformation matrix X is reasonable.

Denote by $\mathbf{1}$ a vector of all ones. In our experiments, we numerically solve linear system (2.1) with a fixed right-hand side $b = \mathbf{1}/||\mathbf{1}||_2$ and compare the residual norms of GMRES, GMRES preconditioned with S^{-1} and TSMRES. The initial approximation for the solution was set as $x_0 = \mathbf{0}_n$, i.e., a vector of all-zeroes, in all of the examples.

Concerning subspaces, with the standard GMRES, approximations are generated from the Krylov subspaces $\mathcal{K}_j(S; b)$, whereas GMRES preconditioned with S^{-1} from the right (here denoted as GMRES-S^{-1}), uses Krylov subspaces of the form $\mathcal{K}_j(S^{-1}; b)$. With TSMRES approximations are generated from the two-sided Krylov subspaces $\mathcal{K}^{\pm}_{\lceil \frac{j}{2} \rceil}(S; b)$. For both the standard and two-sided Krylov subspaces, we have a representation for $I + S$ (or $I + S^{-1}$) in the basis generated by adjoining identity matrix to the upper left corner of the representation obtained for S (or S^{-1}).

To generate cases of interest, we randomize sets of points with a predefined shape in the complex plane. In the following, we denote by $U_k(a, b)$ a set of k random numbers from uniform distribution in the interval (a, b), i.e.,

$$U_k(a, b) = \{x \mid x \in (a, b), P(x) = 1/(b - a)\}, \tag{2.34}$$

where $P(x)$ denotes the probability of x.

Our motivation for using sets of points from the uniform distribution is based on emulating continuous clusters of eigenvalues. By using uniformly distributed sets within some shape, we aim to construct a well-behaving general case. For GMRES it is well known that eigenvalues separated from the rest of the spectrum may slow down convergence [33, 91]. Also, since we

are minimizing a discrete spectrum, it is well-known that the multiplicities of the eigenvalues matter [33]. We note that in [33] it was shown that using random points from the uniform distribution on some interval provides slightly better convergence than using points from the Chebyshev distribution, for instance. The reported difference between the two-distributions was not drastic, though.

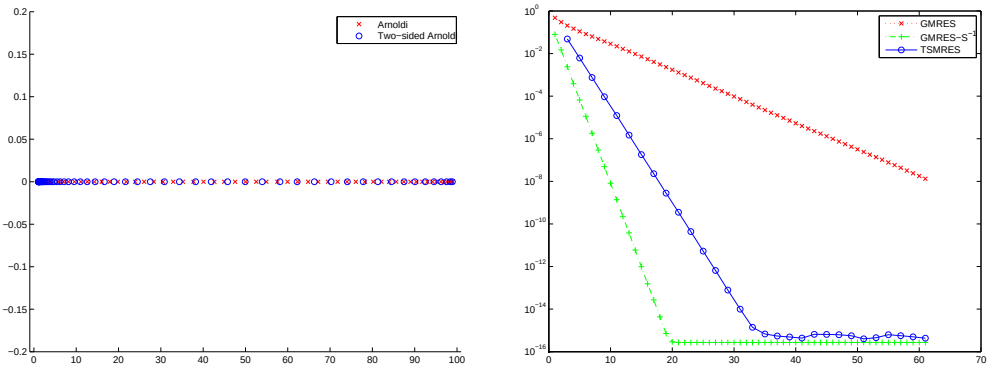
By the convergence theory of GMRES and by Proposition 2.7 for TSMRES, the convergence of the methods is linked to the polynomial approximation problem on the spectrum of S. Therefore, for each test case, we plot the eigenvalues approximated with the Arnoldi and the two-sided Arnoldi methods.

Denote by x_m^{gm}, $x_m^{\mathrm{gm}-S^{-1}}$ and x_m^{ts} the approximate solutions generated by GMRES, GMRES-S^{-1} and TSMRES, respectively. We monitor the convergence of the methods by computing the residual norms $r_m^{\mathrm{gm}} = ||b - (I + S)x_m^{\mathrm{gm}}||_2$, $r_m^{\mathrm{gm}-S^{-1}} = ||b - (I + S)x_m^{\mathrm{gm}-S^{-1}}||_2$ and $r_m^{\mathrm{ts}} = ||b - (I + S)x_m^{\mathrm{ts}}||_2$ from scratch at each step of the iteration. For each test case, we plot the computed residual norms in a logarithmic scale (y-axis) versus the subspace size (x-axis).

First we consider cases where spectrum of S lies on the real axis.

Example 2.6 We choose $\sigma(S)$ to be contained in the interval $(1, 99)$ on the real line. To this end, we set $\sigma(S) = U_n(1, 99)$ with $n = 4000$.

The spectrum approximated with the standard Arnoldi and two-sided Arnoldi for a subspace of size 61 and the convergence of GMRES, GMRES-S^{-1} and TSMRES are shown in Figure 2.2.



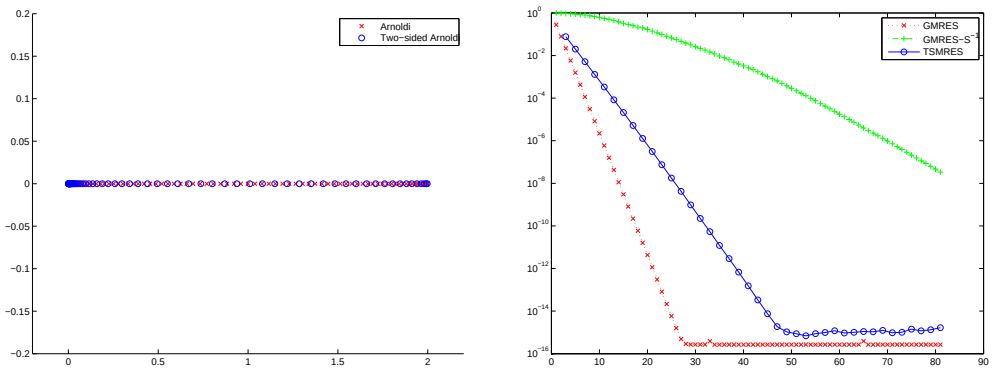
(a) Computed spectrum (b) Residual norm

Figure 2.2: Example 2.6, approximated spectrum and convergence

Since $\sigma(S^{-1})$ is tightly clustered on the interval $(1/99, 1)$, approximation

by polynomials in S^{-1} is efficient and we can expect GMRES-S^{-1} to converge rapidly. On the other hand, since $\sigma(S)$ is not very clustered, accurate approximation requires high degree polynomials in S. However, since the matrix S is symmetric positive definite, minimum residual methods, such as GMRES converge, see [105, Theorem 5.10, pp. 141]. The convergence speed can be slow, however. As expected, TSMRES converges between the two extremes, using a subspace about twice the size of GMRES preconditioned with S^{-1} to achieve a residual of the same magnitude.

Example 2.7 We choose to $\sigma(S)$ be contained in the interval $(0, 2)$ on the real line by setting $\sigma(S) = U_n(0, 2)$ with $n = 4000$. Approximated spectrum for a subspace of size 81 and the convergence of the methods are shown in Figure 2.3.



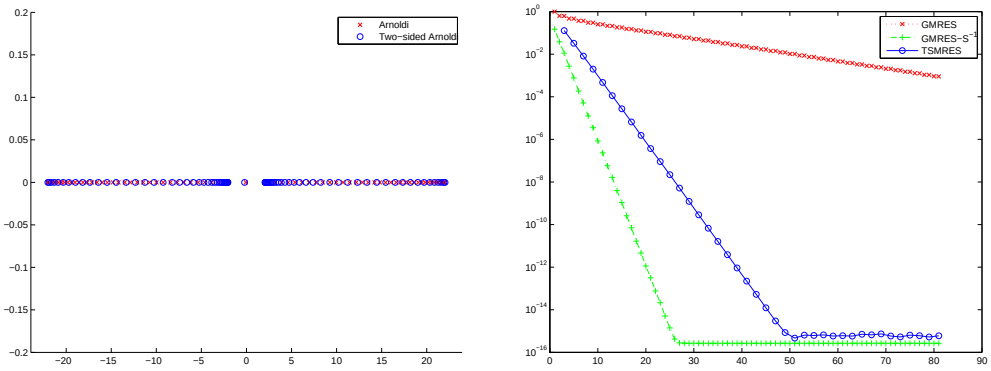
(a) Computed spectrum (b) Residual norm

Figure 2.3: Example 2.7, approximated spectrum and convergence

The spectrum of S is clustered in the interval $(0, 2)$ and thus approximation by polynomials in S is feasible. On the other hand, for S^{-1}, we have $\sigma(S^{-1}) \in \{\lambda \mid \lambda \in (1/2, \infty)\}$, and therefore approximation by polynomials in S^{-1} requires polynomials of high degree. However, as in the previous example, since the matrix S^{-1} is positive definite GMRES-S^{-1} is guaranteed to converge, although slowly. We observe the convergence speed of GMRES-S^{-1} to be much worse than that of the standard GMRES. Again, TSMRES converges at slightly slower pace than GMRES, but faster than GMRES-S^{-1}.

Example 2.8 We let $\sigma(S)$ to contain two isolated regions on the real line with $\sigma(S) \in \{\lambda \mid \lambda \in (-20, -20/19) \cup \lambda \in (20/19, 20)\}$ by setting $\sigma(S) =$

$U_n(-20, -20/19) \cup U_n(20/19, 20)$ with $n = 2000$. Figure 2.4 shows the the convergence and the approximated eigenvalues for a subspace of size 81.



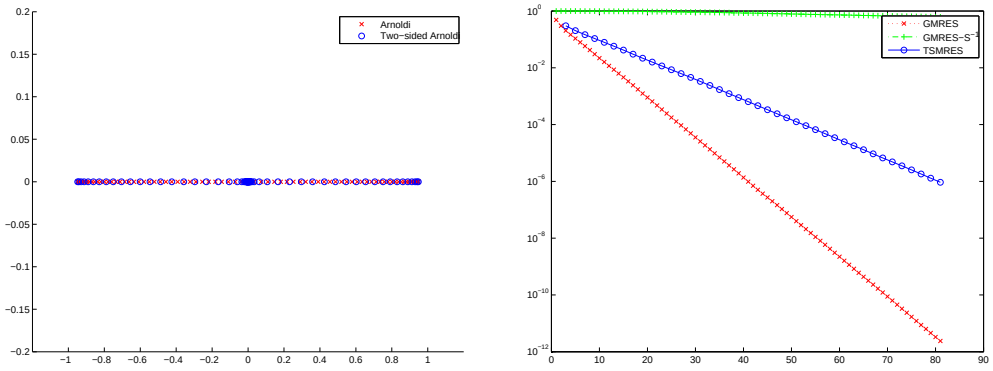
(a) Computed spectrum (b) Residual norm

Figure 2.4: Example 2.8, approximated spectrum and convergence

For the spectrum of S^{-1} we have $\sigma(S^{-1}) \in \{\lambda \mid \lambda \in (-19/20, -1/20) \cup \lambda \in (1/20, 19/20)\}$, so the spectral radius of S^{-1} smaller than that of S. Since the spectrum is on the real line, we can then expect to be able to approximate the solution more efficiently with polynomials in S^{-1} than with polynomials in S. From the numerical experiments we can conclude that for this example the GMRES-S^{-1} method converges much faster than the standard GMRES. In this case the TSMRES method requires approximately two times more iterations than GMRES-S^{-1} to achieve a residual norm of the same magnitude.

Example 2.9 In this example, we set $\sigma(S)$ to contain a line in the proximity of the origin $\sigma(S) \in \{\lambda \mid \lambda \in (-19/20, 19/20),\ \lambda \neq 0\}$ by setting $\sigma(S) = U_n(-19/20, 0) \cup U_n(0, 19/20)$ with $n = 2000$. The approximated spectrum for a subspace of size 81 and the convergence of the methods are shown in Figure 2.9.

The spectrum of S is clustered around the origin with the spectral radius slightly smaller than one, so $\sigma(I+S)$ does not surround the origin. Then the approximation by polynomials in S is efficient and we can expect GMRES to converge. On the other hand, for the spectrum of S^{-1} it holds $\sigma(S^{-1}) \in \{\lambda \mid \lambda \in (-\infty, -20/19) \cup \lambda \in (20/19, \infty)\}$, i.e., the spectrum of S^{-1} is scattered over the real line excluding the interval $(-20/19, 20/19)$. Thus, since $\sigma(I + S^{-1})$ surrounds the origin, the approximation by polynomials in S^{-1} is not efficient and we can expect GMRES-S^{-1} to perform poorly. The TSMRES method requires almost twice as many iterations as GMRES to

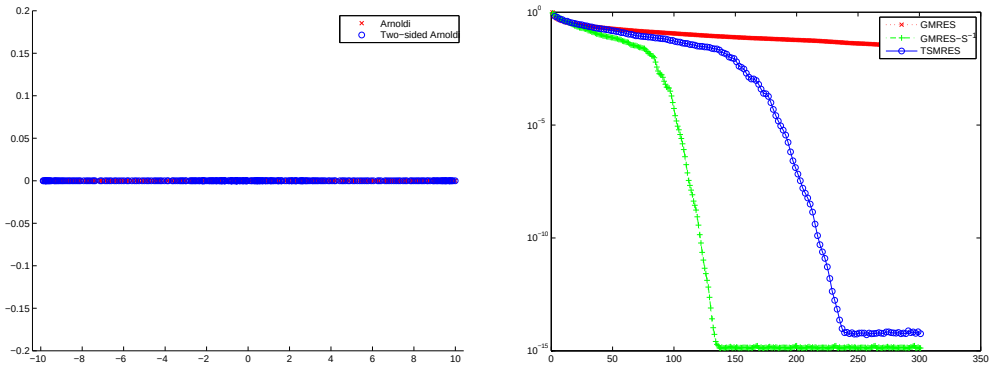


(a) Computed spectrum (b) Residual norm

Figure 2.5: Example 2.9, approximated spectrum and convergence

achieve a similar residual, but still converges much faster than GMRES-S^{-1}.

Example 2.10 In the last example to involve a spectrum on the real line, we construct the spectrum of S as $\sigma(S) \in \{\lambda \mid \lambda \in (-10, 10),\ \lambda \neq 0\}$ by setting $\sigma(S) = U_n(-10, 0) \cup U_n(0, 10)$ with $n = 500$. Figure 2.10 shows the results and the approximated spectrum for a subspace of size 301.



(a) Computed spectrum (b) Residual norm

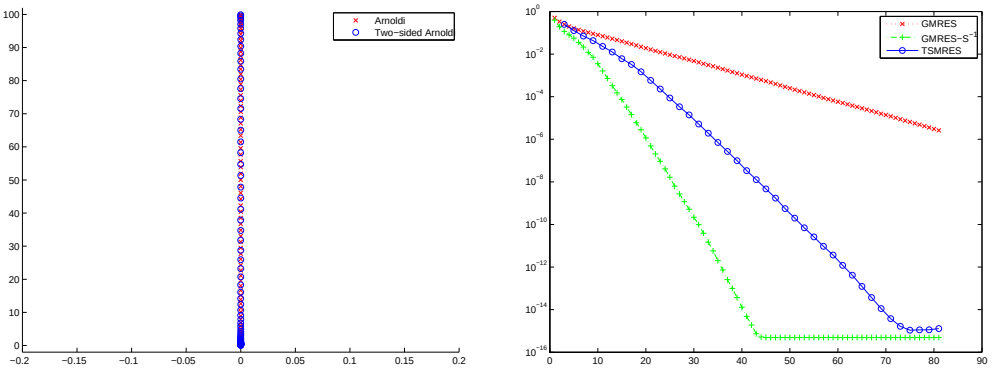
Figure 2.6: Example 2.10, approximated spectrum and convergence

For the spectrum of S^{-1} we have $\sigma(S^{-1}) \in \{\lambda \mid \lambda \in (-\infty, -1/10) \cup \lambda \in (1/10, \infty)\}$. Since the eigenvalues of S are evenly distributed in the interval $(-10, 10)$, most eigenvalues of S^{-1} are clustered in the sets $(-1, -1/10)$ and $(1/10, 1)$. Therefore, once the approximations to the outermost eigenvalues are complete, polynomials in S^{-1} can be efficiently used and we can expect

GMRES-S^{-1} to converge rapidly. On the other hand, the spectrum of S is very scattered and approximation by polynomials in S is difficult. Again, TSMRES converges faster than GMRES, requiring about a third more iterations when compared to GMRES-S^{-1} to achieve a residual norm of the same magnitude.

The next two examples are concerned with cases where the spectrum of S is purely imaginary.

Example 2.11 We construct the spectrum of S as $\sigma(S) \in \{\lambda \mid \lambda \in (i, 100i)\}$ by setting $\sigma(S) = U_n(1, 100)i$ with $n = 4000$. Figure 2.7 shows the results and the approximated spectrum for a subspace of size 81



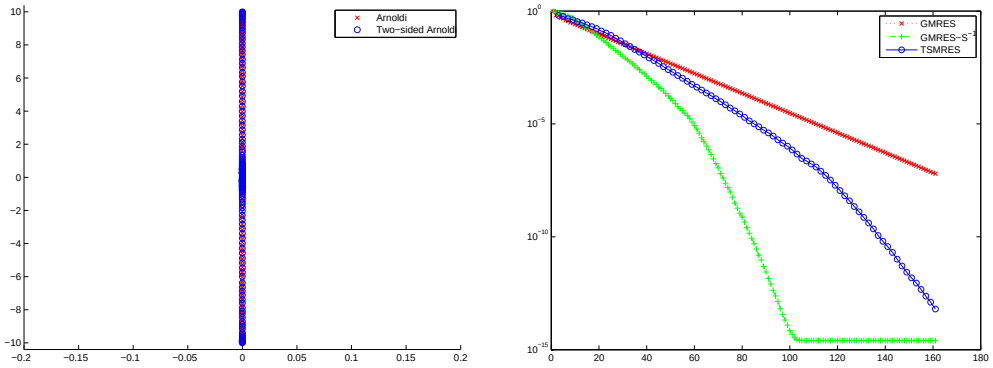
(a) Computed spectrum (b) Residual norm

Figure 2.7: Example 2.11, approximated spectrum and convergence

The conclusions are almost similar to those of Example 2.6. Since $\sigma(S)$ is not tightly clustered and the spectral radius of S is large, high order polynomials in S are required and the convergence speed of GMRES is slow. On the other hand, for the spectrum of S^{-1} we have $\sigma(S^{-1}) \in \{\lambda \mid \lambda \in (-i/100, -i)\}$ and the approximation by low degree polynomials in S^{-1} is feasible. Thus the GMRES-S^{-1} can be expected to converge rapidly. The TSMRES method convergences quite rapidly, requiring about a third more iterations than GMRES-S^{-1} to achieve a residual norm of the same magnitude.

Example 2.12 We construct the spectrum of the matrix S as $\sigma(S) \in \{\lambda \mid \lambda \in (-10i, 0) \cup \lambda \in (0, 10i)\}$ by setting $\sigma(S) = U_n(-10, 0)i \cup U_n(0, 10)i$ with $n = 500$. The spectrum approximated with a subspace of size 161 and the convergence of the methods are shown in Figure 2.8.

The convergence is very similar to that of Example 2.6. For S^{-1} we have $\sigma(S^{-1}) \in \{\lambda \mid \lambda \in (-\infty, -i/10) \cup \lambda \in (i/10, \infty)\}$, but again the most



(a) Computed spectrum (b) Residual norm

Figure 2.8: Example 2.12, approximated spectrum and convergence

eigenvalues of S^{-1} are clustered in the sets $(-i, -i/10)$ and $(i/10, i)$. The convergence of TSMRES is quite similar to that of GMRES-S^{-1}, although almost a third more iterations are required to achieve a residual of the same order of magnitude.

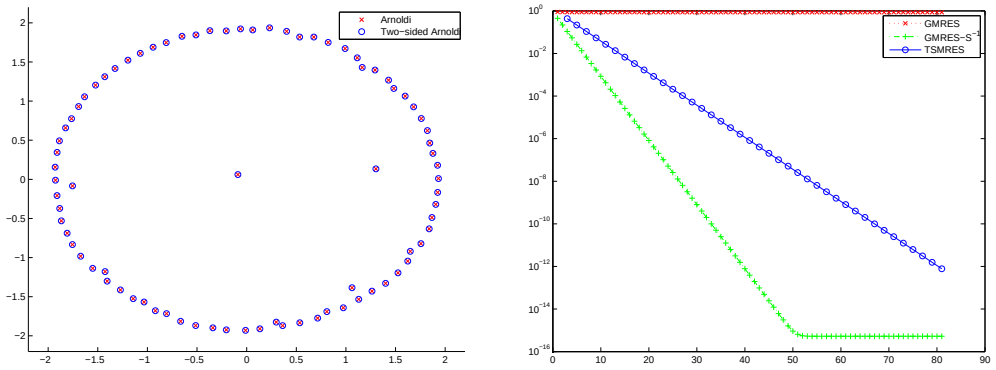
In the following examples, we consider cases where the spectrum of S is on a circle of radius r with a midpoint α in the complex plane. The circle can be parameterized as

$$\{\lambda \mid \lambda = \alpha + re^{i2\pi\theta},\ \theta \in (0,1)\}. \tag{2.35}$$

Example 2.13 We let the spectrum of S to lie on a circle centered at the origin with a radius 2. To this end, we set $\alpha = 0$, $r = 2$, $n = 4000$, $\theta = U_n(0,1)$ and use the parametrization (2.35) to construct $\sigma(S)$. The spectrum approximated with a subspace of size 81 and the convergence of the methods are shown in Figure 2.9.

For S^{-1} we have $\sigma(S^{-1}) \in \{\lambda \mid \lambda \in \frac{1}{2}e^{-i2\pi\theta}\}$, i.e., the spectral radius of S^{-1} is smaller than one. Then $\sigma(I + S^{-1})$ does not surround the origin and approximation by polynomials in S^{-1} can be expected to be successful. Therefore GMRES-S^{-1} can be expected to converge. On the other hand, since $\sigma(I+S)$ surrounds the origin, approximation by low degree polynomials in S is not possible. In practice the GMRES method does not converge before the nth step.

By Theorem 2.10, approximation by a polynomial in both S and S^{-1} is can be expected to be successful. The TSMRES method converges, requiring twice the number of iterations when compared to GMRES-S^{-1} to achieve a residual norm of the same magnitude. Such behaviour is due to the minimal residual property of TSMRES in the two-sided subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; b)$. Since in

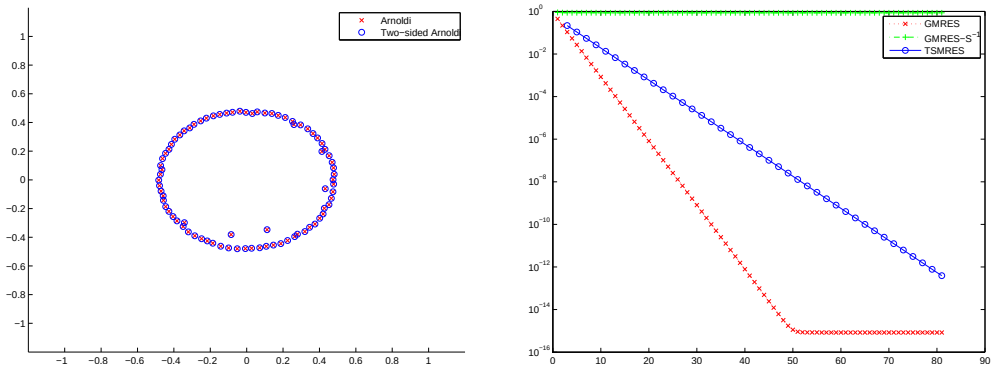


(a) Computed spectrum (b) Residual norm

Figure 2.9: Example 2.13, approximated spectrum and convergence

the approximation problem (2.30), only polynomials in S can be interpreted to be of use, we can expect the convergence speed to be roughly halved when compared to that of GMRES-S^{-1}.

Example 2.14 We now reverse the situation of the previous example, i.e., we let the spectrum of S to lie on a circle centered at the origin with a radius $1/2$. We set $\alpha = 0$, $r = 1/2$, $n = 4000$, $\theta = U_n(0, 1)$ and again use (2.35) to construct $\sigma(S)$. The spectrum approximated with a subspace of size 81 and the convergence of the methods are shown in Figure 2.10.



(a) Computed spectrum (b) Residual norm

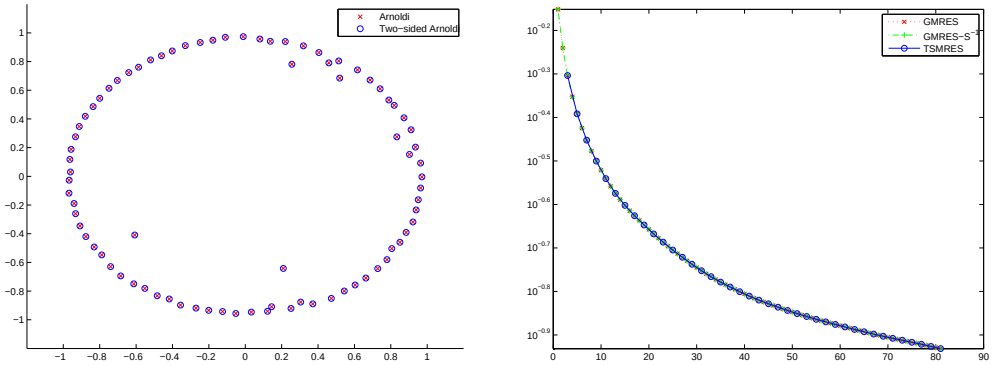
Figure 2.10: Example 2.14, approximated spectrum and convergence

As expected, the situation of Example 2.13 is reversed as suggested by the theory. Since the spectral radius of S is smaller than one, $\sigma(I + S)$ does not surround the origin and an approximation by a low degree polynomials

in S can be expected to be successful. Therefore GMRES can be expected to converge. On the other hand, since $\sigma(I + S^{-1})$ surrounds the origin, approximation by polynomials in S^{-1} cannot be expected to be efficient and GMRES-S^{-1} does not converge. When compared to GMRES, TSMRES requires twice the number of iterations to achieve a residual norm of the same magnitude.

In the extreme case, an approximation with polynomials of low degree is not efficient at all, as shown by the following example.

Example 2.15 We now let the spectrum of S to be on an unit circle. To this end, we set $\alpha = 0$, $r = 1$, $n = 4000$, $\theta = U_n(0, 1)$ and use the equation (2.35) to construct $\sigma(S)$. The spectrum approximated with a subspace of size 81 and the convergence of the methods are shown in Figure 2.11.



(a) Computed spectrum (b) Residual norm

Figure 2.11: Example 2.15, approximated spectrum and convergence

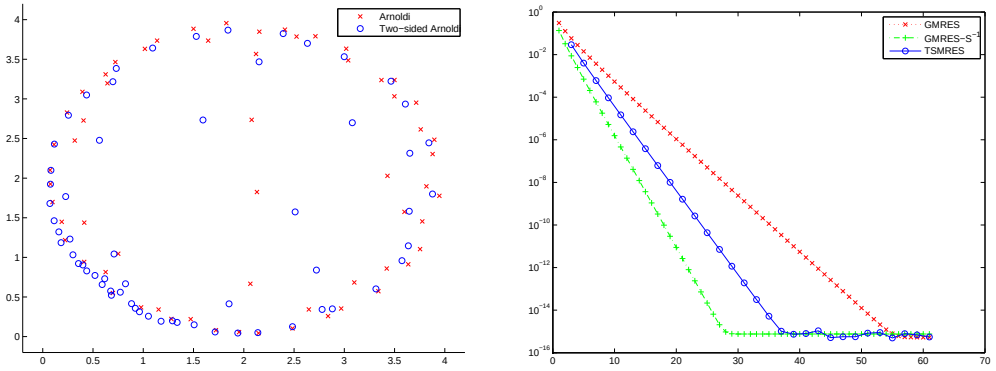
As shown by Figure 2.11, in this case GMRES, GMRES-S^{-1} and TSMRES perform equally poorly and all nearly stagnate. Since both $\sigma(I+S)$ and $\sigma(I+S^{-1})$ nearly surround the origin, polynomials in S or in S^{-1} apparently cannot be efficiently used for approximation. Thus the convergence speed is very slow for all the methods and virtually no convergence to a solution is acquired before the nth step of iteration is reached.

Summarizing the results of Examples 2.13, 2.14 and 2.15 for $r > 1$ GMRES-S^{-1} has the fastest convergence among the three methods. Respectively, for $r < 1$, GMRES is the fastest of the three methods. In the particular examples, the performance of TSMRES is always between GMRES and GMRES-S^{-1}, with $r = 1$ being the special case where all methods give equal results.

We now consider generalizations of the previous examples, i.e., cases where the spectrum of S is a disc (or a set of discs) with a midpoint α and radius r in the complex plane. In the following, we denote by $\gamma(\alpha, r) = \{z \mid |z-\alpha| \leq r\}$ a set of points α inside a disc with a radius r in the complex plane. In the numerical examples, $\sigma(S)$ is constructed to belong to

$$\{\lambda \mid \lambda = p + r_0 e^{i2\pi\theta}, r_0 \in (0, r], \theta \in (0, 1)\}. \tag{2.36}$$

Example 2.16 In this example, we let the spectrum of S to lie on a disc on the upper left half plane with the midpoint $2+2i$ and radius 1. To construct $\sigma(S)$, we use (2.36) with $r = U_n(0,1)$, $\theta = U_n(0,1)$, $\alpha = 2+2i$ and $n = 4000$. The spectrum approximated with a subspace of size 61 and the convergence of the methods are shown in Figure 2.12.



(a) Computed spectrum (b) Residual norm

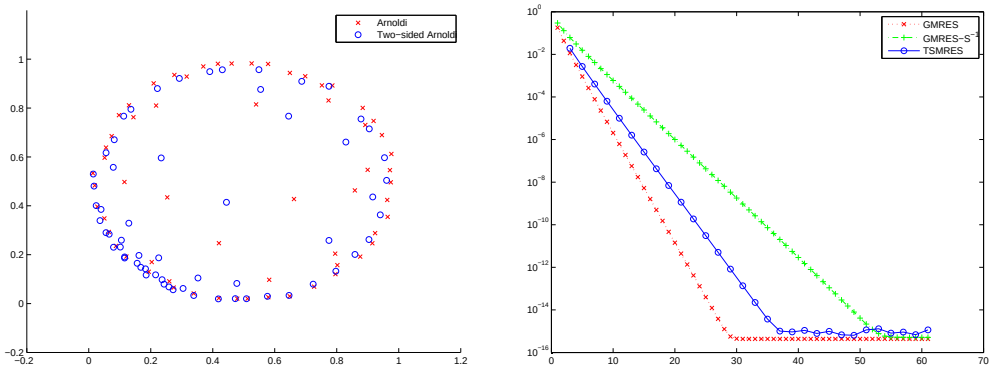
Figure 2.12: Example 2.16, approximated spectrum and convergence

The results are very similar to those of Example 2.6. The matrices S and S^{-1} are positive definite, with both S^{-1} and S having the spectrum that is tightly clustered. Again, by [105, Theorem 5.10, pp. 141], both GMRES and GMRES-S^{-1} can be expected to converge.

For this example, all of the methods converge in very few iterations. The convergence of GMRES-S^{-1} is slightly faster than that of standard GMRES due to better clustering of the eigenvalues of S^{-1} near the origin. The TSMRES method converges a slightly slower than the GMRES-S^{-1} method, requiring a third more iterations to achieve a residual norm of the same magnitude.

Example 2.17 In this example, we let the spectrum of S to lie on a disc on the upper left half plane similarly to the previous example, but modify the

location and the radius of the disc. To construct $\sigma(S)$, we set $r = U_n(0, 1/4)$ and $\theta = U_n(-15/16, 15/16)/2$, $\alpha = 0$ and $n = 4000$ and use the equation (2.36). The spectrum approximated with a subspace of size 61 and the convergence of the methods are shown in Figure 2.13.



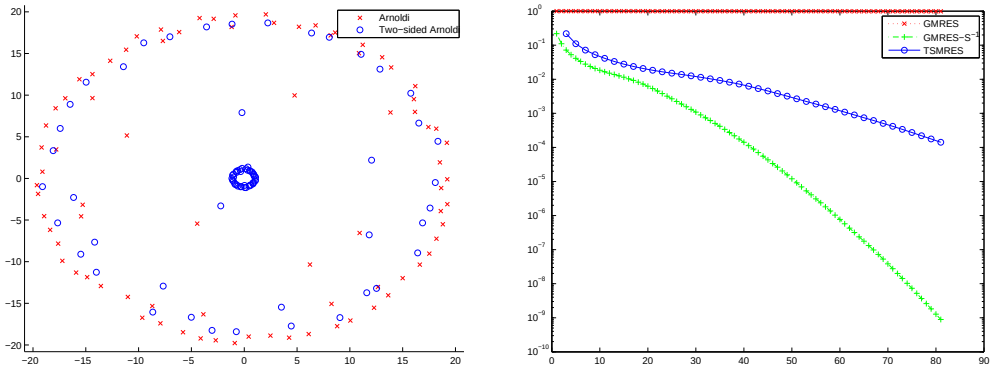
(a) Computed spectrum (b) Residual norm

Figure 2.13: Example 2.17, approximated spectrum and convergence

The situation of the previous example is now reversed, similarly as in Example 2.7 related to Example 2.6. Again, since S is positive definite, both GMRES and GMRES-S^{-1} can be expected to converge. As expected, in this example, the standard GMRES is the fastest method. TSMRES is slightly slower than GMRES, requiring a third more iterations to achieve a residual norm of the same magnitude.

Example 2.18 We now consider an extension of Example 2.13 by setting the spectrum of S to form a ring around the origin, i.e., $\sigma(S) \in \{\lambda \in \sigma(S) \mid \lambda \in \{\gamma(p_0, r_2) \setminus \gamma(p_0, r_1)\}\}$. In order to construct $\sigma(S)$, we use (2.36) with $r_0 = (r_1, r_2)$ by choosing $r_0 = U_n(r_1, r_2)$ and $\theta = U_n(0, 1)$ with $\alpha = 0$, $r_1 = 1$, $r_2 = 20$, and $n = 4000$. The spectrum approximated with a subspace of size 81 and the convergence of the methods are shown in Figure 2.14.

Since $\sigma(I + S)$ surrounds the origin, polynomials in S cannot be expected to be used efficiently. Therefore this example is very difficult for the standard GMRES, which nearly stagnates. On the other hand, $\sigma(I + S^{-1})$ does not surround the origin and the approximation by polynomials in S^{-1} can be expected to be efficient. Therefore GMRES-S^{-1} can be expected to converge. Since polynomials in S are not very useful, the TSMRES method requires twice the number of iterations to achieve a residual norm of the same magnitude, when compared to GMRES-S^{-1}. Therefore the situation is analogous to that of Example 2.13.

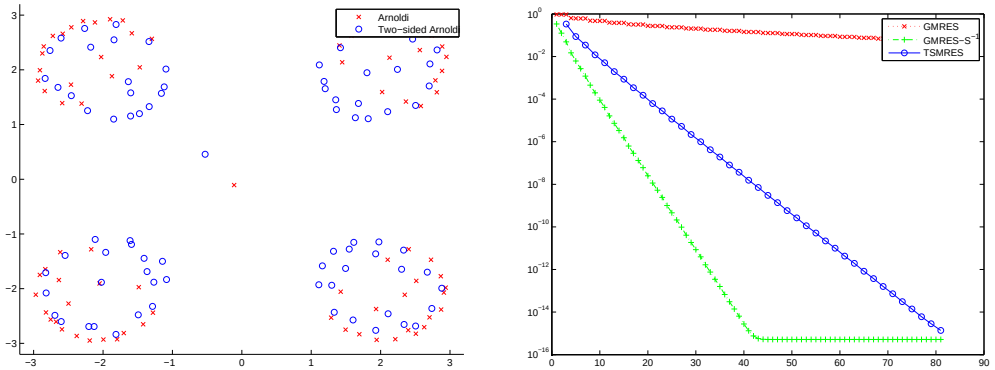


(a) Computed spectrum (b) Residual norm

Figure 2.14: Example 2.18, approximated spectrum and convergence

Example 2.19 We construct the spectrum of S to belong to a set of discs symmetrically aligned about the origin in each quadrant of the complex plane, i.e., $\sigma(S) \in \{\lambda \in \sigma(S) \mid \lambda \in \gamma(\alpha_1, r) \cup \lambda \in \gamma(\alpha_2, r) \cup \lambda \in \gamma(\alpha_3, r) \lambda \in \gamma(\alpha_4, r)\}$. For each disc, we set radius $r = 1$ and choose the midpoints of the discs as $\alpha_1 = 2 + 2i$, $\alpha_2 = -2 + 2i$, $\alpha_3 = -2 - 2i$, $\alpha_4 = 2 - 2i$. Using (2.36) with $r_0 = U_n(0, r)$, $\theta = U_n(0, 1)$ and $n = 1000$ we form $\sigma(S)$ as the union of the four discs of eigenvalues.

The spectrum approximated with a subspace of size 81 and the convergence of the methods are shown in Figure 2.15.



(a) Computed spectrum (b) Residual norm

Figure 2.15: Example 2.19, approximated spectrum and convergence

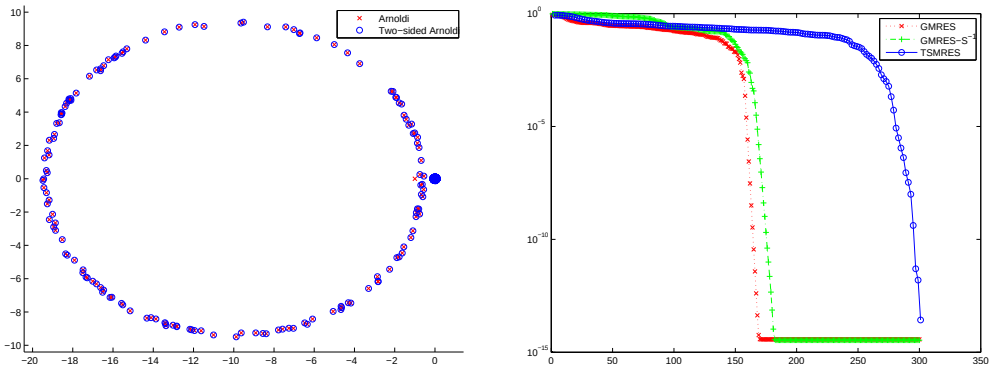
Approximation by polynomials in S is apparently not efficient due to the scattering of the eigenvalues and thus standard GMRES converges very slowly . On the other hand, $\sigma(S^{-1})$ is clustered in discs around points

$\pm 1/2 \pm 1/2$ with radii of approximately $2/7$, so $\sigma(I + S^{-1})$ does not surround the origin and therefore approximation by polynomials in S^{-1} can be expected to be efficient. The GMRES-S^{-1} method converges quite rapidly. The TSMRES method is slightly slower, requiring about twice the number of iterations when compared to GMRES-S^{-1} to achieve a residual norm of the same magnitude.

In the following example, we consider a case which is very difficult for the two-sided Krylov subspaces.

Example 2.20 We construct the spectrum of S to form two complex rings, one surrounding the point $\alpha_1 = -10$ with an internal radius $r_1 = 925/100$ and an external radius $r_2 = 950/100$ and the other surrounding origin at $\alpha_2 = 0$ with an internal radius $r_3 = 100/950$ and an external radius $r_4 = 100/925$. We construct the rings as in Example 2.18, i.e., by using (2.36) with $r_0 = U_n(r_1, r_2)$ or $r_0 = U_n(r_3, r_4)$, $\theta = U_n(0, 1)$ and $n = 150$.

Denote by $\sigma_1(S)$ the spectrum of the ring surrounding point α_1 and by $\sigma_2(S)$ the spectrum of the ring surrounding point α_2. Then the the spectrum of S equals the union of two rings as $\sigma(S) = \sigma_1(S) \cup \sigma_2(S)$. The spectrum approximated with a subspace of size 301 and the convergence of the methods are shown in Figure 2.16.



(a) Computed spectrum (b) Residual norm

Figure 2.16: Example 2.20

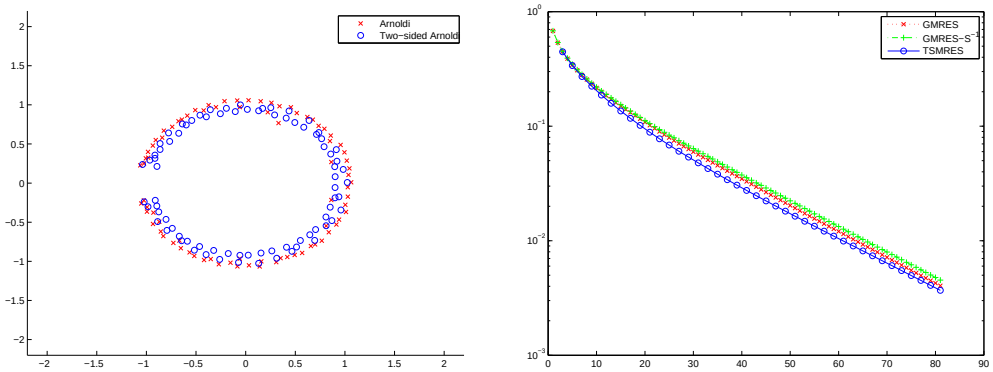
The polynomials in S can be efficiently used to approximate $\sigma_2(S)$ but not $\sigma_1(S)$. On the other hand, the polynomials in S^{-1} can be efficiently used to approximate $\sigma_1(S)$ but not $\sigma_2(S)$. Therefore, after an approximation to $\sigma_2(S)$ (or $\sigma_1(S)$) has been found, the GMRES method (or the GMRES-S^{-1} method) can be expected to converge rapidly. From the numerical results we can observe that once the size of subspace has exceeded 150, i.e., the size of

the complex rings $\sigma_1(S)$ and $\sigma_2(S)$, both GMRES and GMRES-S^{-1} converge rapidly.

The use of the two-sided subspaces for approximation of $\sigma(S)$ is difficult since neither of the polynomials in S or in S^{-1} can be efficiently used for approximation. With the two-sided subspaces, the degree of the polynomials in S and in S^{-1} equals the current dimension of the subspace divided by two. Due to the construction of $\sigma(S)$, approximation with the two-sided subspaces can be expected only to be efficient when the size of the subspace has nearly reached the dimension of $\sigma(S)$. The behaviour is verified by the numerical results, where we observe that the TSMRES method achieves very little reduction of the residual until the size of subspace is has nearly reached the dimension of the problem.

In the final numerical example, we consider a case where the residual norm of TSMRES is smaller than those of GMRES and GMRES-S^{-1}.

Example 2.21 We construct the spectrum of S to form a ring around the origin having an internal radius $r_1 = 9/10$, an external radius $r_2 = 11/10$ with a part having angle greater than $15\pi/16$ or smaller than $-15\pi/16$ cut off. We construct the spectrum by using the equation (2.36) with $r_0 = U_n(r_1, r_2)$, $\theta = U_n(-15/16, 15/16)/2$ and $n = 4000$. The resulting spectrum for a subspace of size 81 and the convergence of the methods are shown in Figure 2.17.



(a) Computed spectrum (b) Residual norm

Figure 2.17: Example 2.21

Both $\sigma(I + S)$ and $\sigma(I + S^{-1})$ nearly surround the origin. Thus the spectrum is almost equally difficult to approximate with polynomials in S and in S^{-1}. Thereby both GMRES and GMRES-S^{-1} can be expected to slowly converge.

From the numerical results we can observe that the residual norm of TSMRES is slightly smaller than the residual of *both* GMRES *and* GMRES-S^{-1}, especially when the size of the subspace is small. We attribute this behaviour to the structure of $\sigma(S)$ and the way the two-sided subspaces locate eigenvalues. With two-sided subspaces, both large and small eigenvalues are searched from the spectrum simultaneously. In this example such behaviour is beneficiary since the spectrum of S has components on each side of the unit circle.

2.7 Conclusions

In this chapter, we have considered the construction and analysis of the two-sided Krylov subspaces, which can be considered an extension to the standard Krylov-subspaces constructed by using the Arnoldi method. We also introduced a minimum residual method using the two-sided Krylov subspaces, called TSMRES and studied some of its properties both theoretically and numerically.

To study the convergence of TSMRES, we related the two-sided Krylov subspaces $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ to the standard Krylov subspaces $\mathcal{K}_{m+2}(S; q_1)$ and $\mathcal{K}_{m+2}(S^{-1}; q_1)$. Theoretical results were verified by using numerical experiments. We presented numerical examples where approximation properties of the two-sided were either better of worse than those of the standard Krylov subspaces.

In Chapter 3, we consider further generalizations of Krylov subspaces. There we further extended the ideas presented in this chapter by allowing more elements to be added in the Krylov subspaces.

In this chapter we have not considered the numerical implementation of TSMRES at all. The topic was purposefully delayed because it requires careful consideration. Numerically stable implementation of TSMRES is presented in Chapter 4.

Chapter 3

Generalized Krylov subspaces

3.1 Introduction

In this chapter, we consider a method that is a generalization of the method for constructing two-sided Krylov subspaces $\mathcal{K}_j^{\pm}(S; q_1)$ defined in (2.2). The generalization is based on the observation that if S^{-1} is replaced with a matrix which commutes with S, a matrix function of S for instance, we obtain a linearly growing recurrence.

We define generalized Krylov subspaces as

$$\mathcal{K}_J(S; T; q_1) = \operatorname*{span}_{\wp \in \mathbb{P}_j} \{\wp(S, T) q_1\}, \tag{3.1}$$

where $J = (j+1)(j+2)/2$ and $\mathbb{P}_j$ denotes the set of polynomials in two variables of degree j at most and the matrices $S \in \mathbb{C}^{n\times n}$ and $T \in \mathbb{C}^{n\times n}$ have the property $ST = TS$, i.e., S and T commute. By commutativity we can write $\wp(S, T)$ as

$$\begin{aligned} \wp(S,T) &= \sum_{k=0}^{j} \sum_{l=0}^{k} \alpha_{k-l,l} S^{k-l} T^l \\ &= \alpha_{0,0} I + (\alpha_{1,0} S + \alpha_{0,1} T) + (\alpha_{2,0} S^2 + \alpha_{1,1} ST + \alpha_{0,2} T^2) \\ &\quad + (\alpha_{3,0} S^3 + \alpha_{2,1} S^2 T + \alpha_{1,2} S T^2 + \alpha_{0,3} T^3) + \cdots , \end{aligned} \tag{3.2}$$

with coefficients $\alpha_{i,j} \in \mathbb{C}$.

To increase the degree of the polynomial $\wp$ of degree j by one we need to add $j+1$ new terms to the resulting new polynomial. A generalized Krylov subspace with a polynomial degree $j+1$ is thus $\mathcal{K}_{J^+}(S, T; q_1)$ with $J^+ = (m+2)(m+3)/2$, i.e., we have a linearly growing recurrence. In [70], Huhtanen and Larsen consider constructing polynomials in both N and N^*

for a normal matrix N. For the case where S and T do not commute, there exists even more general formulations of Krylov subspaces [4, 65, 83]. Also, in Hoffnung et. al. [65], the commutative case $ST = TS$ is considered a special case of a more a general algorithm, but no explicit formulation is given. The derivation and algorithms given in this chapter were discovered independently, however.

Denote by $J_- = j(j+1)/2$ and $J = (j+1)(j+2)/2$ the number of generated basis vectors after polynomials up to degree $j-1$ and j have been generated, respectively. This is what can be expected generically. Assuming that the matrix S is computationally more inexpensive to apply than T, we device a Gram-Schmidt–type process for constructing an orthonormal basis $\{q_1, q_2, \ldots, q_J\}$ for the generalized Krylov subspace $\mathcal{K}_j(S, T; q_1)$. We then have

$$SQ_{J_-} = Q_J \bar{H}_{J_-},$$

where $Q_J \in \mathbb{C}^{n\times J}$ has orthonormal columns spanning (3.1) and $\bar{H}_J \in \mathbb{C}^{J\times J_-}$ has a block Hessenberg structure with subdiagonal blocks of linearly increasing size.

The generalized Krylov-subspaces $\mathcal{K}_J(S, T; q_1)$ relate to the two-sided subspaces $\mathcal{K}_j^{\pm}(S; q_1)$ considered earlier. Setting $T = S^{-1}$, for the polynomial (3.2) of the generalized subspace $\mathcal{K}_J(S; S^{-1}; q_1)$, we have

$$\begin{aligned} \wp(S, S^{-1}) &= \sum_{k=0}^{j}\sum_{l=0}^{k} \alpha_{k-l,l} S^{k-l} S^{-l} \\ &= \alpha_{0,0} I + (\alpha_{1,0} S + \alpha_{0,1} S^{-1}) + (\alpha_{2,0} S^2 + \alpha_{1,1} I + \alpha_{0,2} S^{-2}) \\ &\quad + (\alpha_{3,0} S^3 + \alpha_{2,1} S + \alpha_{1,2} S^{-1} + \alpha_{0,3} S^{-3}) + \cdots, \end{aligned}$$

from which it then follows, for the span of vectors $\wp(S, S^{-1})q_1$,

$$\operatorname*{span}_{\wp\in\mathbb{P}_j}\{\wp(S, S^{-1})q_1\} = \operatorname*{span}_{\wp,\mathfrak{q}\in\mathbb{P}_j}\{\wp(S)q_1, \mathfrak{q}(S^{-1})q_1\} = \mathcal{K}_j^{\pm}(S; q_1). \tag{3.3}$$

Thus the generalized Krylov subspace $\mathcal{K}_J(S; T; q_1)$ can be considered as a generalization of the two-sided subspace $\mathcal{K}_j^{\pm}(S; q_1)$, generating the two-sided subspace as a special case with $T = S^{-1}$.

The generalized Krylov subspaces can be constructed for any matrices S and T having the commutative property $ST = TS$. When given a matrix S, valid choices of T are also analytic functions f in the domain of the matrix S. For such f we have

$$Sf(S) = f(S)S,$$

i.e., S and $f(S)$ commute. For example, the matrix T can be chosen as the matrix exponential function $T = \exp(S) = \sum_{j=0}^{\infty} S^j/j!$. Of course, such a function may not be computationally inexpensively available.

In the following sections, we describe the construction of an orthonormal basis for the generalized Krylov subspace $\mathcal{K}_J(S;T;q_1)$ and consider its properties. Finally, we present some applications for the generalized Krylov subspaces.

3.2 Orthonormal bases for generalized Krylov subspaces

In this section, we describe a method for the construction of an orthonormal basis $\{q_1.q_2,\ldots,q_M\}$ for the generalized Krylov subspace

$$\mathcal{K}_M(S;T;q_1) = \operatorname*{span}_{\wp\in\mathbb{P}_m}\{\wp(S,T)q_1\},$$

where $M = (m+1)(m+2)/2$, $\mathbb{P}_m$ denotes the set of polynomials in two variables of degree m at most and $\wp(S,T)$ is defined by the formula (3.2). We assume that matrix-vector products with S are computationally more inexpensive than matrix-vector products with T and formulate our algorithms accordingly. At the kth step, we perform $k-1$ multiplications with operator S and apply T once. After performing the related orthogonalizations, we obtain k new basis vectors. Thus we obtain a linearly growing recurrence.

Let m be the maximum degree of the set of polynomials in two variables in the generalized Krylov subspace. We begin by choosing q_1 with $||q_1||_2 = 1$. After this, we compute

$$\hat{q}_2 = Sq_1 - (Sq_1, q_1)q_1,$$

and normalize to have a new basis vector as $q_2 = \hat{q}_2/||\hat{q}_2||_2$. By rearranging the definition of q_2 we have $Sq_1 = \sum_{j=1}^{2} h_{j,1}q_j$, with $h_{1,1} = (Sq_1, q_1)$ and $h_{2,1} = ||\hat{q}_2||_2$.

Then we compute

$$\hat{q}_3 = Tq_1 - (Tq_1, q_1)q_1 - (Tq_1, q_2)q_2,$$

and normalize to have a new basis vector as $q_3 = \hat{q}_3/||\hat{q}_3||_2$. Note that with the above computations, we have acquired two new basis vectors, i.e., computed step $k = 2$ to have three orthonormal vectors spanning $\mathcal{K}_3(S;T;q_1)$.

We continue the construction of the orthonormal basis for $\mathcal{K}_M(S;T;q_1)$ by computing the basis vectors of degree 2, i.e., the vectors associated with the terms $\alpha_{2,0}S^2$, $\alpha_{1,1}ST$ and $\alpha_{0,2}T^2$ in (3.2). To this end, we compute

$$\hat{q}_4 = Sq_2 - (Sq_2, q_1)q_1 - (Sq_2, q_2)q_2 - (Sq_2, q_3)q_3$$

and

$$\hat{q}_5 = Sq_3 - (Sq_3, q_1)q_1 - (Sq_3, q_2)q_2 - (Sq_3, q_3)q_3 - (Sq_3, q_4)q_4,$$

to have, after normalization, two new basis vectors $q_4 = \hat{q}_4/||\hat{q}_4||_2$ and $q_5 = \hat{q}_5/||\hat{q}_5||_2$. In addition, with q_3 we compute

$$\hat{q}_6 = Tq_3 - (Tq_3, q_1)q_1 - (Tq_3, q_2)q_2 - (Tq_3, q_3)q_3 - (Tq_3, q_4)q_4 - (Tq_3, q_5)q_5,$$

and normalize to have $q_6 = \hat{q}_6/||\hat{q}_6||_2$. Again by rearranging the definitions for q_4 and q_5 we have $Sq_2 = \sum_{j=1}^{4} h_{j,2}q_j$ and with $h_{j,2} = (Sq_2, q_j)$, for $j = 1, \ldots, 3$ and $h_{4,2} = ||\hat{q}_4||_2$. Similarly, rearranging the definition for q_5 yields $Sq_3 = \sum_{j=1}^{5} h_{j,3}q_j$ with $h_{j,3} = (Sq_3, q_j)$, for $j = 1, \ldots, 4$ and $h_{5,2} = ||\hat{q}_5||_2$

We now generalize the construction for the kth step. At the step $2 \leq k \leq m+1$, we compute the basis vectors of degree $k-1$, i.e., the vectors associated with the terms $\alpha_{k-l,l}S^{k-l}T^l$, $l = 0, \ldots, k-1$ in (3.2). To this end, we need to perform $k-1$ multiplications with S and one multiplication with T to obtain k new basis vectors of degree $k-1$ in total.

Let $k_0 = k(k-1)/2$ and $k_1 = k(k+1)/2$. We compute

$$\hat{q}_{k_0+l} = Sq_{k_0-(k-1)+l} - \sum_{j=1}^{k_0+l-1} h_{j,k_0-(k-1)+l}q_j, \ l = 1, \ldots, k-1, \tag{3.4}$$

where we denote

$$h_{j,k_0-(k-1)+l} = (Sq_{k_0-(k-1)+l}, q_j), \quad l = 1, \ldots, k-1. \tag{3.5}$$

After normalization we have $k-1$ new basis vectors $q_{k_0+l} = \hat{q}_{k_0+l}/||\hat{q}_{k_0+l}||_2$ and the relations $Sq_{k_0-(k-1)+l} = \sum_{j=1}^{k_0+l} h_{j,k_0-(k-1)+l}q_j$, with

$$h_{k_0+l,k_0-(k-1)+l} = ||\hat{q}_{k_0+l}||_2, \quad l = 1, \ldots, k-1. \tag{3.6}$$

In addition, we have

$$\hat{q}_{k_1} = Tq_{k_0} - \sum_{j=1}^{k_1-1} (Sq_{k_0}, q_j)q_j, \tag{3.7}$$

which yields $q_{k_1} = \frac{\hat{q}_{k_1}}{||\hat{q}_{k_1}||_2}$.

By the relations (3.4) and (3.7) we obtain a linearly growing recurrence, with $M = \sum_{k=1}^{m+1} k = (m+1)(m+2)/2$ basis vectors in total, assuming we do not have any linear dependencies during the construction. We note that the structure of the set $\{q_1, q_2, \ldots, q_M\}$, generated with the relations (3.4) and (3.7), is formally shown in Proposition 3.1.

Denote by $M_- = \sum_{k=1}^{m} k = m(m+1)/2$ the number of basis vectors generated after the mth step. After the step $k = m+1$ has been completed, by (3.4) we have a matrix relation

$$SQ_{M_-} = \begin{bmatrix} Q_{M_-} & q_{M_-+1} & \cdots & q_{M_-+m} \end{bmatrix} \tilde{H}_{M_-}, \tag{3.8}$$

where $Q_{M_-} \in \mathbb{C}^{n \times M_-}$ has orthonormal columns spanning $\mathcal{K}_{M_-}(S; T; q_1)$ and $\tilde{H}_{M_-} \in \mathbb{C}^{M_-+m \times M_-}$ has a block Hessenberg structure. The matrix $\tilde{H}_{M_-}$ contains subdiagonal blocks of linearly increasing size. The nonzero structure of $\tilde{H}_{M_-}$ is shown in (3.9), where $X_{j\times k}$ denotes a j-by-k nonzero block, $\tilde{X}_{j\times k}$ a j-by-k nonzero block with a Hessenberg structure and $\mathbf{0}$ an all-zero block of an appropriate size.

$$\bar{H}_{M_-} = \begin{bmatrix} \tilde{X}_{2\times 1} & X_{2\times 2} & X_{2\times 3} & \cdots & X_{2\times m} \\ \mathbf{0} & \tilde{X}_{3\times 2} & X_{3\times 3} & \cdots & X_{3\times m} \\ \vdots & \mathbf{0} & \tilde{X}_{4\times 3} & \cdots & \vdots \\ \vdots & \ddots & \mathbf{0} & \ddots & \vdots \\ \mathbf{0} & \cdots & \cdots & \mathbf{0} & \tilde{X}_{(m+1)\times m} \end{bmatrix} \tag{3.9}$$

After the relation (3.8) has been acquired, to have representation of S of the form (1.13) in the basis generated, it remains to compute inner products of the form

$$h_{k,j} = (Sq_k, q_j), \tag{3.10}$$

for $k = M_- + 1, \ldots, M$ and $j = 1, \ldots, M$. We then have a representation of S in the basis generated for $\mathcal{K}_M(S; T; q_1)$ as

$$SQ_M = Q_M \bar{H}_M, \tag{3.11}$$

where the values of the nonzero entries $h_{i,j}$ of $\bar{H}_M \in \mathbb{C}^{M\times M}$ are determined by the relations (3.5), (3.6) and (3.10). The nonzero structure of $\bar{H}_M$ is of the form $\begin{bmatrix} H_{M_-} & \bar{H}_{M,m} \end{bmatrix}$ where $\bar{H}_{M,m}$ is a full M-by-$(m+1)$ matrix and

$$H_{M_-} = \begin{bmatrix} \tilde{H}_{M_-} \\ \mathbf{0} \end{bmatrix},$$

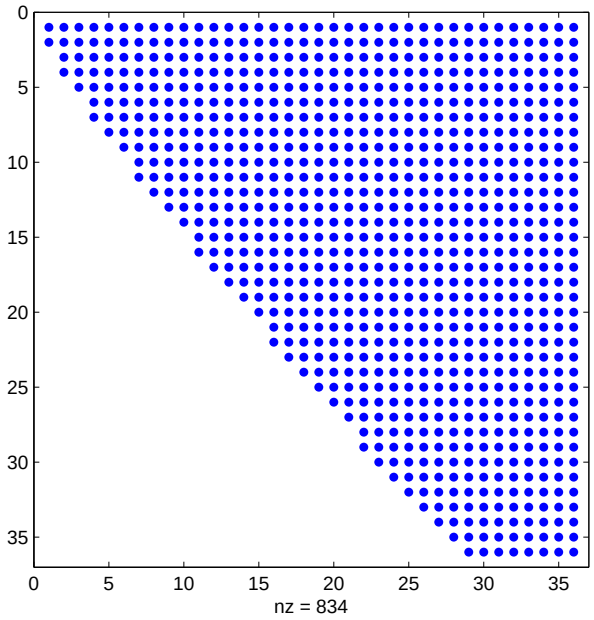


Figure 3.1: The nonzero structure of the matrix $\bar{H}_{36}$

where the nonzero structure of $\tilde{H}_{M_-}$ defined as in (3.9) and $\mathbf{0}$ denotes an all-zero matrix of the size 1-by-M_-. For an illustration, Figure 3.1 shows the nonzero structure of the matrix $\bar{H}_M$ with $M = 36$, i.e., $m = 7$.

In the implementation, we use the iterated modified Gram Schmidt process to orthogonalize the generated basis vectors. Denote by u the machine precision. By Hoffmann [64], see also [15, 26, 99], we have Algorithm 3.1 to orthonormalize a vector w against an orthonormal basis $[q_1, q_2, \ldots, q_k]$. The algorithm, which we call ORTHOGON, constructs a vector q_{k+1} and coefficients $h_j = (w, q_j)$, $j = 1, \ldots, k+1$ such that $q_{k+1} = \hat{q}_{k+1}/||\hat{q}_{k+1}||_2$ with $\hat{q}_{k+1} = w - \sum_{j=1}^{k} h_j q_j$ holds, i.e., the computed q_{k+1} vector is a numerically orthogonal projection to the complement of $\text{span}\{q_1, \ldots, q_k\}$.

In Algorithm 3.1, the parameter κ needs to be chosen positive and such that the condition $\kappa > (0.83 - u)^{-1}$ holds. In our implementation, we set $\kappa = \sqrt{2}$. For other choices of κ, see [64] and references therein.

The relations (3.4) and (3.7) describe a construction of the orthonormal set of basis vectors $\{q_1, q_2, \ldots, q_M\}$ for the generalized Krylov subspace $\mathcal{K}_M(S; T; q_1)$. From the construction, after the remaining inner products have been computed, we also obtain a representation of the form (3.11). We interpret the given construction as a generalized Arnoldi process for commuting

Algorithm 3.1 $[q_{k+1}, [h_1, h_2, \ldots, h_{k+1}]]$=ORTHOGON$([q_1, \ldots, q_k], w)$

1: $\omega_0 = ||w||_2$
2: **repeat**
3: $\quad \omega = ||w||_2$
4: $\quad$ **for** $j = 1, \ldots, k$ **do**
5: $\quad\quad s_j = (w, q_j)$
6: $\quad\quad w = w - s_j q_j$
7: $\quad\quad h_j = h_j + s_j$
8: $\quad$ **end for**
9: $\quad$ **if** $||w||_2 < 2nu\omega_0$ **then**
10: $\quad\quad h_{k+1} = 0$
11: $\quad\quad q_{k+1} = 0$
12: $\quad\quad$ **return**
13: $\quad$ **end if**
14: **until** $||w||_2 > \omega/\kappa$
15: $h_{k+1} = ||w||_2$
16: $q_{k+1} = w/h_{j+1}$

operators S and T. Using ORTHOGON, we have the generalized Arnoldi process as Algorithm 3.2.

To keep Algorithm 3.2 simple, we have assumed that the construction does not break down before the $(m+1)$th step has been completed, i.e., the basis vectors q_{k_0+l}, $l = 1, \ldots, k-1$ and q_{k_1} can always be generated on lines 7 and 10. Denote by k_S and k_T the indices of the first of those vectors which cannot be generated on lines 7 and 10, respectively. Then the set of vectors $\{q_1, \ldots, q_{k_S-1}, Sq_{k_S-(k-1)}\}$ (or $\{q_1, \ldots, q_{k_T-1}, Tq_{k_T-k}\}$) is linearly dependent and the construction continues from the next index which produces a new linearly independent basis vector. This complicates the indexing somewhat since in order to generate a set of new vectors, we have to keep track of the vectors which have been successfully generated in the previous iteration round.

Denote by N the amount of the basis vectors generated. For clarity, we now present a more general version of Algorithm 3.2 by using index sets for the generated basis vectors. Denote by $\mathcal{Q}_S$ and $\mathcal{Q}_T$ the sets containing the indices of the basis vectors generated in the $(k-1)$th iteration round after multiplications by S and T, respectively. Similarly, denote by $\mathcal{Q}_S^N$ and $\mathcal{Q}_T^N$ the sets containing indices of the basis vectors of degree $k-1$, generated on the kth iteration round.

In the kth iteration round we generate a set of new basis vectors of degree $k-1$ by multiplying the basis vectors with the indices in the set $\mathcal{Q}_S \bigcup \mathcal{Q}_T$

Algorithm 3.2 Generalized Arnoldi process, simple version

Choose q_1 such that $||q_1||_2 = 1$
for $k = 2, \ldots, m+1$ **do**
  $k_0 = k(k-1)/2$, $k_1 = k(k+1)/2$
  for $l = 1, \ldots, k-1$ **do**
    $j = k_0 - (k-1) + l$
    $w = Sq_j$
    $[q_{k_0+l}, [h_{1,j}, \ldots, h_{k+1,j}]]$ =ORTHOGON$([q_1, \ldots, q_{k_0+l-1}], w)$
  end for
  $w = Tq_{k_0}$
  $[q_{k_1}, [t_{1,k_1}, \ldots, t_{k+1,k_1}]]$ =ORTHOGON$([q_1, \ldots, q_{k_1-1}], w)$
end for
$M = (m+1)(m+2)/2$
for $l = 1, \ldots, m+1$ **do**
  $j = M - (m+1) + j$
  $w = Sq_j$
  $[q_t, [h_{1,j}, \ldots, h_{M,j}, h_{M+1,j}]]$ =ORTHOGON$([q_1, \ldots, q_M], w)$
end for

by S and the basis vectors with the indices in the set $\mathcal{Q}_T$ by T. After multiplications with S, we augment the set $\mathcal{Q}_S^N$ to contain the indices of the new basis vectors generated. Similarly, after a multiplication with T, if a new basis vector is generated, we augment the set $\mathcal{Q}_T^N$ to contain the index of that vector. Then finally, after the $(m+1)$th step has been completed, we compute the inner products of the form (Sq_k, q_j), where $j = 1, \ldots, N$ and $k \in \mathcal{Q}_S \bigcup \mathcal{Q}_T$. As a result, we have Algorithm 3.3.

We note that during the iteration, the sets $\mathcal{Q}_S$ and $\mathcal{Q}_S^N$ can contain at most m elements, respectively. Similarly, the sets $\mathcal{Q}_T$ and $\mathcal{Q}_T^N$ can contain at most one element. If no breakdowns occur in the generation of the basis vectors, Algorithm 3.3 is equivalent to Algorithm 3.2, i.e., it generates exactly the same set $\{q_1, \ldots, q_N\}$ with $N = (m+1)(m+2)/2$. If some of the basis vectors cannot be generated, the recursion obtained for the basis vectors does not grow linearly and the nonzero structure of the matrix H_M may be difficult to determine beforehand.

In the following section, we consider the theoretical properties of the generalized Arnoldi process, i.e., Algorithm 3.2. We also consider how the generalized Arnoldi process relates to the two-sided Arnoldi process 2.1 given in Section 2.2 both in an exact and finite precision arithmetic.

Algorithm 3.3 Generalized Arnoldi process

Choose q_1 such that $||q_1||_2 = 1$
$N = 1$, $\mathcal{Q}_S = \mathcal{Q}_T = \{1\}$, $\mathcal{Q}_S^N = \mathcal{Q}_T^N = \emptyset$
for $k = 2, \ldots, m+1$ **do**
  if $\mathcal{Q}_S \bigcup \mathcal{Q}_T = \emptyset$ **then**
    return
  end if
  for $\forall j \in \mathcal{Q}_S \bigcup \mathcal{Q}_T$ **do**
    $w = Sq_j$
    $[q_t, [h_{1,j}, \ldots, h_{N,j}, h_{N+1,j}]]$ =ORTHOGON$([q_1, \ldots, q_N], w)$
    if $h_{N+1,j} > 0$ **then**
      $N = N + 1$, $q_N = q_t$, $\mathcal{Q}_S^N = \mathcal{Q}_S^N \bigcup \{N\}$
    end if
  end for
  if $\mathcal{Q}_T \neq \emptyset$ **then**
    Set j equal to first element of $\mathcal{Q}_T$
    $w = Tq_j$
    $[q_t, [t_1, \ldots, t_{N+1}]]$ =ORTHOGON$([q_1, \ldots, q_N], w)$
    if $t_{N+1} > 0$ **then**
      $N = N + 1$, $q_N = q_t$, $\mathcal{Q}_T^N = \{N\}$
    end if
  end if
  $\mathcal{Q}_S = \mathcal{Q}_S^N$, $\mathcal{Q}_T = \mathcal{Q}_T^N$, $\mathcal{Q}_S^N = \mathcal{Q}_T^N = \emptyset$
end for
for $\forall j \in \mathcal{Q}_S \bigcup \mathcal{Q}_T$ **do**
  $w = Sq_j$
  $[q_t, [h_{1,j}, \ldots, h_{N,j}, t_{N+1}]]$ =ORTHOGON$([q_1, \ldots, q_N], w)$
end for

3.3 Properties of generalized Krylov subspaces

In this section, we describe some properties of Algorithm 3.2. We begin with a generalization of Proposition 2.1, describing the structure of the generated basis vectors.

Proposition 3.1. *Assume Algorithm 3.2 does not break down before the* $(m+1)$*th step and let* $M = (m+1)(m+2)/2$ *be the number of basis vectors generated. Then, the set* $\{q_1, q_2, \ldots, q_M\}$ *is an orthonormal basis of the*

generalized Krylov subspace

$$\mathcal{K}_M(S;T;q_1) = \operatorname*{span}_{\wp \in \mathbb{P}_m} \{\wp(S,T)q_1\},$$

where $M = (m+1)(m+2)/2$ *and* $\mathbb{P}_m$ *denotes the set of polynomials in two variables of degree* m *at most and the polynomial* $\wp(S,T)$ *is defined as in* (3.2).

Proof. The proof is a straightforward generalization of the proof of Proposition 2.1. Again, the generated set of vectors is orthonormal by construction. It remains to show that the generated set forms a basis for the generalized Krylov subspace $\mathcal{K}_M(S;T;q_1)$. We show the claim by induction.

For $j = 1$, the inner loop on lines from 2 to 11 in Algorithm 3.2 runs through once. After applying S and computing the related orthonormalizations, we have $q_2 = \hat{q}_2/||\hat{q}_2||_2$ with $\hat{q}_2 = Sq_1 - (Sq_1, q_1)q_1$. Thus for q_2 is holds $q_2 \in \operatorname{span}\{q_1, Sq_1\}$. Similarly, after applying T, we have $q_3 = \hat{q}_3/||\hat{q}_3||_2$ with $\hat{q}_3 = Tq_1 - (Tq_1, q_1)q_1 - (Tq_1, q_2)q_1$. Thus for q_3 it holds $q_3 \in \operatorname{span}\{q_1, Sq_1Tq_1\}$. Thus it follows that $\operatorname{span}\{q_1, q_2, q_3\} = \mathcal{K}_3(S;T;q_1)$ and the claim holds for $j = 1$.

We now assume that the claim holds up to $j = m - 1$. In the next step, with $j = m$, by the relations (3.4) and (3.7), we compute $m - 1$ new basis vectors associated with the matrix S and a single basis vector associated with the matrix T.

With $j = m$ we have $k_0 = m(m+1)/2$ and $k_1 = (m+1)(m+2)/2$. By the induction assumption, it holds $q_{k_0-m+l} \in \mathcal{K}_{M_-}(S;T;q_1)$ where $l = 1, \ldots, m$ and $M_- = m(m+1)/2$. By the relation (3.4), after generating $m - 1$ new basis vectors associated with the matrix S, the span of the vectors $\{q_1, \ldots, q_{k_0}, q_{k_0+1}, \ldots, q_{k_0+m}\}$ equals $\operatorname{span}_{\wp \in \tilde{\mathbb{P}}_m}\{\wp(S,T)q_1\}$, where $\tilde{\mathbb{P}}_m$ denotes the set of polynomials in two variables such that the degree of the first variable is m and the degree of the second variable is $(m-1)$ at most. After generating the basis vector associated with the matrix T, we have $\{q_1, \ldots, q_{k_0}, q_{k_0+1}, \ldots, q_{k_0+m}, q_{k_1}\} \in \operatorname{span}_{\wp \in \mathbb{P}_m}\{\wp(S,T)q_1\} = \mathcal{K}_M(S;T;q_1)$ and the claim follows. □

Since the matrices S and T commute, they share the same invariant subspace. Define the set of eigenvectors of T as $V = \{v \in \mathbb{C}^n |\ Tv = \mu v\}$. If S and T commute, for $v \in V$ we have

$$T(Sv) = S(Tv) = S(\mu v) = \mu(Sv),$$

and it follows that $Sv \in V$, i.e., V is an invariant subspace of S. By induction it then follows that there exists unitary matrix Q such that $QSQ^* = R_S$ and $QTQ^* = R_T$ with R_S and R_T both upper triangular.

For the case when Algorithm 3.3 terminates with $k < m+1$, we have the following proposition.

Proposition 3.2. *When Algorithm 3.3 terminates with $k < m+1$ after generating N vectors, the generated set $\{q_1, \ldots, q_N\}$ is an invariant subspace of both S and T.*

Proof. Denote by $\mathcal{Q}_S$, $\mathcal{Q}_T$, $\mathcal{Q}_S^N$ and $\mathcal{Q}_T^N$ the index sets of the $(k-1)$th iteration round. When Algorithm 3.3 terminates on the kth round having $k < m+1$, the condition on line 4 holds, i.e., sets $\mathcal{Q}_S^N$ and $\mathcal{Q}_T^N$ must both be empty, i.e., no new basis vectors have been generated during the $(k-1)$th iteration round. On the other hand, the sets $\mathcal{Q}_S$ and $\mathcal{Q}_T$ must both be nonempty, or else the algorithm would have stopped already in the beginning of the $(k-1)$th iteration round.

Since no new basis vectors were generated during the $(k-1)$th iteration round, all the vectors Sq_j with the indices $j \in \mathcal{Q}_S \bigcup \mathcal{Q}_T$ must be linearly dependent in the set $\{q_1, \ldots, q_N\}$. By the relation (3.4), we then have

$$Sq_j = \sum_{l=1}^{N} h_{l,j} q_l,$$

with $h_{l,j} = (Sq_j, q_l)$ for all $j \in \mathcal{Q}_S \bigcup \mathcal{Q}_T$. Similarly, by the relation (3.7), for the vector Tq_j with index $j \in \mathcal{Q}_T$ we have

$$Tq_j = \sum_{l=1}^{N} t_l q_l,$$

with $t_l = (Tq_j, q_l)$ for $j \in \mathcal{Q}_T$. Thus, the set $\{q_1, \ldots, q_N\}$ is an invariant subspace for both S and T. □

Our interest into generalized Krylov subspaces arises from the relation (3.3). When Algorithm 3.3 is used with $T = S^{-1}$ up to the $(j+1)$th step, the two-sided subspace $\mathcal{K}_j^{\pm}(S; q_1)$ is generated. We now assume exact arithmetic and that Algorithm 2.1 does not break down before the jth step. By Lemma 2.5 we have that in Algorithm 3.3, only two new basis vectors can be generated each iteration round, i.e., the length of the recursion is two. After computing the $(j+1)$th step of Algorithm 3.3, we have $SQ_{2j+1} = Q_{2j+2}\bar{H}_{2j+1}$, where Q_{2j+1} denotes the matrix having orthonormal columns spanning $\mathcal{K}_j^{\pm}(S; q_1)$ and $\bar{H}_{2j+1}$ is a block upper Hessenberg matrix with 2-by-2 subdiagonal blocks. This essentially proves the following proposition.

Proposition 3.3. *Assume exact arithmetic and that Algorithm 2.1 does not break down before the mth step. Then, with* $T = S^{-1}$*, Algorithm 3.3 generates a subspace* $\mathcal{K}_j^{\pm}(S; q_1)$ *with relation* $SQ_{2j+1} = Q_{2j+2}\bar{H}_{2j+1}$ *where* Q_{2j+1} *and* $\bar{H}_{2j+1}$ *are defined as in Proposition*(2.3).

In finite precision arithmetic we cannot expect the condition $SS^{-1} = I$ to hold, especially if the matrix S is very ill-conditioned. Therefore, with $T = S^{-1}$, Algorithm 3.2 may generate more than two new basis vectors each iteration round in practice and a linearly growing recursion is obtained. Since in finite precision arithmetic the condition $SS^{-1} = S^{-1}S$ is often much less strict than $SS^{-1} = I$, Algorithm 3.2 can be used to generate the basis of the two-sided Krylov subspace $\mathcal{K}_j^{\pm}(S; q_1)$ in a numerically stable way. For finite precision error analysis, we refer to [61].

In the previous sections we have considered the generalized Krylov subspace $\mathcal{K}_M(S; T; q_1)$ merely as an extension of the two-sided Krylov subspace $\mathcal{K}_m^{\pm}(S; q_1)$. In the following section, we present cases where the generalized Krylov subspaces can be used.

3.4 Applications for generalized Krylov subspaces

We now briefly consider applications, where the generalized Krylov subspaces can be used. Denote by f an analytic function in an open subset containing the spectrum of A. We consider the computation of

$$u = f(A)v, \tag{3.12}$$

where the matrix A is large and sparse. As an example of such a function, we consider the matrix exponential function, defined as

$$\exp(S) = I + S + \frac{S^2}{2!} + \cdots = \sum_{j=0}^{\infty} S^j/j!. \tag{3.13}$$

The matrix exponential function appears, for instance, in solving the initial value problems

$$x' = Ax, \quad x(t_0) = x_0$$

with a large and sparse constant coefficient matrix $A \in \mathbb{C}^{n\times n}$ and an initial condition x_0. The solution vector x at time t is given by

$$x = \exp(tA)x_0,$$

i.e., only a matrix-vector product of the form

$$\exp(tA)v, \tag{3.14}$$

for a vector $v \in \mathbb{C}^n$ is needed.

For a more general example, consider exponential integration methods for large nonlinear initial value problems

$$x' = f(x), \quad x(t_0) = x_0, \tag{3.15}$$

with a a nonlinear function $f(x)$ and an initial condition x_0. The solution of (3.15) can be approximated with the exponentially fitted Euler method [63], written as

$$x_1 = x_0 + h\varphi(hA)f(x_0),$$

where h is the step size, $A = f'(x_0)$ is the Jacobian matrix and

$$\varphi(z) = \frac{\exp(z) - 1}{z}. \tag{3.16}$$

Again, only a matrix-vector product of the form

$$\varphi(hA)v, \tag{3.17}$$

for a vector $v \in \mathbb{C}^n$ is needed. Since the matrices $\exp(tA)$ and $\varphi(hA)$ are generally full even when the matrix A is sparse, we seek a method for computing (3.14) or (3.17) without computing the matrices $\exp(tA)$ or $\varphi(hA)$.

Let us now consider an approximation of (3.12) in the Krylov subspace $\mathcal{K}_m(A; v_1) = \text{span}\{v_1, Av_1 \ldots, A^{m-1}v_1\}$ with $v_1 = v/||v||_2$. After performing m steps of the Arnoldi method, i.e., Algorithm 1.1, we have

$$AV_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T = V_{m+1} \bar{H}_m,$$

where, as before, V_m has orthonormal columns spanning $\mathcal{K}_m(A; v_1)$, H_m is an m-by-m upper Hessenberg and e_m denotes the mth column of an m-by-m identity matrix. Therefore the product $f(A)v$ is approximated in the Krylov subspace $\mathcal{K}_m(A; v_1)$ as

$$u_m = ||v||_2 V_m f(H_m) e_1, \tag{3.18}$$

where e_1 denotes the first column of an m-by-m identity matrix. For instance, for the matrix exponential function we have

$$\exp(A)v \approx ||v||_2 V_m \exp(H_m) e_1.$$

Thus we have reduced the computation of a large sparse matrix exponential into a small full one. The small full matrix exponential can be computed with direct methods, such as Padé approximation combined with scaling and squaring, see survey [86] and references therein. Similar techniques can be used to have approximations for $\varphi(hA)v$ by computing the values of $\varphi(hH_m)$, see [63].

For the product $\exp(A)v$, good approximations can be often found for a relatively small m [100]. Preconditioning of the matrix exponential function is studied, among others, by Eshof and Hochbruck [119] and by Castillo and Saad in [21]. In the case of a general matrix function, such as (3.16) or the matrix square root, using Krylov subspaces generated with S and S^{-1} can lead to an improved convergence [35]. For an additional analysis and applications, see Knizhnerman [78], Hochbruck and Lubich [62], Hochbruck, Lubich and Selhofer [63] and Beckermann and Reichel [5].

Another frequent application for the generalized Krylov subspaces is the solution of quadratic eigenproblems. In the quadratic eigenproblem one seeks scalars λ with vectors $x, y \in \mathbb{C}^n$ such that

$$(\lambda^2 M - \lambda D - K)x = 0, \quad y^*(\lambda^2 M - \lambda D - K) = 0, \tag{3.19}$$

with the matrices $M, D, K \in \mathbb{C}^{n\times n}$ holds. Vectors x and y are called the right and the left eigenvectors, corresponding to the quadratic eigenvalues λ. Note that the quadratic eigenproblem (3.19) may have up to $2n$ eigenvalues and eigenvectors [86].

The use of the generalized Krylov subspace for solving (3.19) in the case of noncommuting matrices M, D, K and nonsingular M has been proposed by Bai and Su in [4]. In their approach, the quadratic eigenproblem (3.19) is reduced to

$$(\theta^2 M_m + \theta D_m + K_m)g = 0,$$

with

$$M_m = Q_m^T M Q_m, \;\; D_m = Q_m^T D Q_m, \;\; , K_m = Q_m^T K Q_m,$$

where $Q_m \in \mathbb{C}^{n\times m}$ denotes a matrix containing orthonormal basis for the generalized subspace $\mathcal{K}_m(-M^{-1}D; -M^{-1}K; u)$ as its columns. The small quadratic eigenproblem (3.4) for the pairs (θ, z) with $z = Q_m g$ is then solved in two steps. First, a transformation into generalized eigenproblem of the form

$$Cy = \lambda G y,$$

is made by setting $y^T = [\lambda g^T \; g^T]$ and

$$C = \begin{bmatrix} -D_m & -K_m \\ I_m & \mathbf{0} \end{bmatrix}, \quad G = \begin{bmatrix} M_m & \mathbf{0} \\ \mathbf{0} & I_m \end{bmatrix}.$$

Then the generalized eigenproblem is solved with the QZ-algorithm [55]. For similar approaches see Li and Ye [83] and Hoffnung, Li and Ye [65].

In the context of the quadratic eigenvalue problem, the generalized Krylov subspaces presented in this chapter can be used if the matrices $M^{-1}D$ and $M^{-1}K$ commute. An example of such a case arises in vibrational acoustics with proportional damping, see [27, Chapter 11.9] and references therein.

3.5 Conclusions

In this chapter we have considered a generalization of the two-sided Krylov subspaces $\mathcal{K}_m^{\pm}(S; q_1)$ in terms of subspaces $\mathcal{K}_M(S; T; q_1)$ where the matrices S and T commute. It was shown that with such a generalized Krylov subspace we acquire a linearly growing recurrence.

We also considered some potential applications for the generalized Krylov subspaces. These were shown to include equations involving matrix exponentials (3.14), matrix functions (3.17) and solving quadratic eigenvalue problems (3.19). Although our interest in the generalized Krylov subspaces stems from the numerical computation of the two-sided subspaces, a potential use for Algorithm 3.3 could also be computation of approximations to the products of the form (3.12).

With the choice $T = S^{-1}$, Algorithm 3.2 can be used to generate a basis for a two-sided Krylov subspace and a representation of the matrix S in the basis generated in a numerically stable way. When Algorithm 2.1 is used to generate the basis for the two-sided Krylov subspace $\mathcal{K}_m(S; q_1)$, $2m + 1$ basis vectors are generated assuming no breakdown occurs before the mth step. Due to the linearly growing recurrence, in finite precision arithmetic, the basis generated for $\mathcal{K}_M(S; q_1)$ with Algorithm 3.2 can contain up to $(m+1)(m+2)/2$ basis vectors. To generate and store the additional basis vectors, Algorithm 3.2 requires more computations and more storage when compared to using Algorithm 2.1. Therefore, the generation of the two-sided subspaces $\mathcal{K}_m^{\pm}(S; q_1)$ with Algorithm 3.2 is not computationally efficient. In the next Chapter, we consider the numerical computation of the two-sided Krylov subspaces used by TSMRES without resorting to a computation and storage of additional basis vectors.

Chapter 4

Implementation of TSMRES

4.1 Introduction

In this chapter, we consider a computationally efficient and numerically stable implementation of TSMRES. Both of the subjects are extremely important for real-life computations and therefore require a careful consideration.

We begin by considering efficient solution of the least-squares problem (2.20), i.e., $\min_{y_{m+2}\in\mathbb{C}^{m+2}} ||\beta e_1 - (\bar{I} + \bar{H}_{k+2})y||_2$. We show how to solve (2.20) by using Givens rotations to transform the block Hessenberg matrix $\bar{I} + \bar{H}_{k+2}$ into an upper triangular form. It is also shown how to efficiently compute a representation of the matrix S in the basis generated for the two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ without performing additional orthogonalizations or matrix-vector products.

After considering the efficient implementation, we describe a numerically stabilized implementation of TSMRES. The new implementation is mathematically equivalent to the implementation of TSMRES given in Chapter 2, but by our numerical experiments seems to be numerically more stable. We then analyze the computational complexity and memory consumption of different implementations of TSMRES and compare the results obtained to those of GMRES. It is shown, that for subspaces of similar size, the computational effort is of equivalent magnitude.

In the end of this chapter, we consider the numerical stability of TSMRES. Some consideration is given why the numerically stabilized implementation of TSMRES, given in Section 4.4, seems to be numerically more stable than the implementation given in Section 2.3. By our experiments this is linked with the phenomenon that in a finite precision arithmetic, a numerically accurate representation of the vectors Sq_{k+2} with $k = 1, \ldots, m$ in the basis generated for $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ cannot be computed by using only the first $k + 3$

basis vectors.

We begin this chapter by making some observations on how to implement TSMRES efficiently. For completeness, we begin by reviewing how the Givens rotations are defined.

4.2 Givens rotations

Givens rotations, sometimes also referred as Jacobi rotations, are orthogonal transformations commonly used for transforming elements of a matrix into zeroes. They were originally used by Jacobi in solving symmetric eigenvalue problems [55, 72]. In this section, we follow the definition for numerically accurate computation of the Givens rotations as given in [12, 13].

We define a Givens rotation $G_{j,k}$ as an unitary n-by-n matrix which only acts on the components j and k of a given vector v. Then the Givens rotation $G_{j,k}$ is an identity matrix, except for the part affecting jth and kth components, i.e., the entries $g_{j,j}$, $g_{j,k}$, $g_{k,j}$ and $g_{k,k}$. Restricting to this 2-by-2 part we have

$$\begin{pmatrix} g_{j,j} & g_{j,k} \\ g_{k,j} & g_{k,k} \end{pmatrix} \begin{pmatrix} v_j \\ v_k \end{pmatrix} = \begin{pmatrix} c_{j,k} & s_{j,k} \\ -\bar{s}_{j,k} & c_{j,k} \end{pmatrix} \begin{pmatrix} v_j \\ v_k \end{pmatrix} = \begin{pmatrix} r \\ 0 \end{pmatrix}, \tag{4.1}$$

where v_j denotes the jth component the vector v. Since $G_{j,k}$ is unitary, we have

$$G_{j,k} G_{j,k}^* = I,$$

from which we have the conditions

$$|c_{j,k}|^2 + |s_{j,k}|^2 = 1, \quad c_{j,k} - \bar{c}_{j,k} = 0. \tag{4.2}$$

By (4.2) we have the restriction $c_{j,k} \in \mathbb{R}$. For real and positive entries v_j and v_k, it is common to choose $c_{j,k}$ and $s_{j,k}$ as

$$\begin{aligned} c_{j,k} &= \frac{v_j}{\sqrt{v_j^2 + v_k^2}}, \\ s_{j,k} &= \frac{v_k}{\sqrt{v_j^2 + v_k^2}}, \\ r &= \sqrt{v_j^2 + v_k^2}, \end{aligned}$$

even though also negative values of $c_{j,k}$, $s_{j,k}$ and r would also satisfy the conditions (4.2). If either or both of v_j and v_k are complex, the conditions (4.2) are not sufficient to determine $c_{j,k}$, $s_{j,k}$ and r uniquely. Requiring the

mapping from (v_j, v_k) to $(c_{j,k}, s_{j,k}, r)$ to be continuous whenever possible, for $v_j, v_k \neq 0$ we have

$$\begin{aligned} c_{j,k} &= \frac{|v_j|}{\sqrt{|v_j|^2 + |v_k|^2}}, \\ s_{j,k} &= \text{sign}(v_j)\frac{\bar{v}_k}{\sqrt{|v_j|^2 + |v_k|^2}}, \\ r &= \text{sign}(v_j)\sqrt{|v_j|^2 + |v_k|^2}, \end{aligned} \tag{4.3}$$

where $\text{sign}(x)$ denotes the sign function, defined for $x \in \mathbb{C}$ as

$$\text{sign}(x) = \begin{cases} x/|x|, & |x| \neq 0, \\ 1, & x = 0. \end{cases} \tag{4.4}$$

When $v_k = 0$ (including the case $v_j = 0$), again by continuity, the relations (4.3) simplify to $c_{j,k} = 1$, $s_{j,k} = 0$ and $r = v_j$. Finally, with $v_j = 0$ but $v_k \neq 0$, we choose $c_{j,k}$ and $s_{j,k}$ to be continuous as v_j approaches zero from the right, which yields $c_{j,k} = 0$, $s_{j,k} = \text{sign}(\bar{v}_k)$ and $r = |v_k|$.

Ignoring the special cases, when properly implemented, the computational complexity for computing a single Givens rotation readily follows from (4.3) as 7 floating point operations plus a single square root, i.e., $O(1)$.

4.3 Practical implementation considerations

We now consider the implementation of Algorithm 2.2 in practice. Efficient means to solve the least-squares problem (2.20) and to monitor the residual norm $||b - (I + S)x_m||_2$ of the approximation x_m are presented. We also consider how to efficiently compute a representation of the matrix S in the basis generated, i.e., how to compute the columns h_{k+2} of $\bar{H}_{m+2}$ for $k = 1, 3, \ldots, m$, without performing any additional orthogonalizations or matrix-vector products.

A practical difficulty in Algorithm 2.2 is that it does not provide either the approximate solution x_k or the norm of the residual vector $||b - (I + S)x_k||_2$ explicitly at each step. Therefore it is not easy to know when to stop.

A technique commonly used in the GMRES method to solve the GMRES minimization problem (1.16), is to transform the upper Hessenberg matrix into an upper triangular form by using plane rotations. This allows the norm of the residual to be monitored and enables an efficient way solve the GMRES minimization problem (1.16). For a reference, see for example [105, Chapter 6.5.3, pp.167]. In what follows, we present a similar technique for TSMRES.

Consider solving the TSMRES minimization problem (2.20), that is, $\min_{y_{m+2}\in\mathbb{C}^{m+2}} ||\beta e_1-(\bar{I}+\bar{H}_{m+2})y_{m+2}||_2$. Since the matrix $\bar{H}_{m+2}$ has a Hessenberg-like structure with 2-by-2 subdiagonal blocks, to transform it into an upper triangular form we need to eliminate entries at the positions $(j+3, j+1)$, $(j+2, j+1)$ and $(j+3, j+2)$, for $j = 1, 3, \ldots, m$ odd. This is accomplished in $O(m^2)$ operations by using plane rotations.

Assume that m steps of Algorithm 2.2 have been performed. For the jth step of the iteration with $j = 1, 3, \ldots, m$ odd, we define the product of the Givens rotation matrices $G_{j,k} \in \mathbb{C}^{(m+3)\times(m+3)}$ as

$$\Omega_j = G_{j+3,j+2}G_{j+2,j+1}G_{j+3,j+1}, \tag{4.5}$$

where $G_{j,k}$ denotes a rotation to clear the entry at position (j, k). After denoting by $\mathcal{G}_m = \Omega_m\Omega_{m-2}\ldots\Omega_1 G_{2,1}$ the product of matrices to transform $\bar{H}_{k+2}$ into upper triangular form, we have

$$\begin{aligned} \bar{R}_{m+2} &= \mathcal{G}_m(\bar{I} + \bar{H}_{m+2}), & (4.6)\\ \bar{g}_{m+2} &= \mathcal{G}_m(\beta e_1) = (\gamma_1, \ldots, \gamma_{m+3})^T, & (4.7) \end{aligned}$$

where $\bar{R}_{m+2}$ is an upper triangular $(m + 3)$-by-$(m + 2)$ matrix with the last row containing only zeroes.

An important observation considering the practical implementation of rotations $\mathcal{G}_m$ is that $G_{k+3,k+1}$ commutes with Ω_j for all $j < k$. This allows us to write

$$\begin{aligned} \mathcal{G}_m &= \Omega_m\Omega_{m-2}\cdots\Omega_1 \\ &= G_{m+3,m+2}G_{m+2,m+1}G_{m+3,m+1} \\ &\quad G_{m+1,m}G_{m,m-1}G_{m+1,m-1}\cdots \\ &\quad G_{4,3}G_{3,2}G_{4,2}G_{2,1} \\ &= G_{m+3,m+2}G_{m+2,m+1}G_{m+1,m}G_{m,m-1}\cdots \\ &\quad G_{4,3}G_{3,2}G_{2,1}G_{m+3,m+1}G_{m+1,m-1}\cdots G(4,2), \end{aligned}$$

by which the rotations to clear entries at positions $(j + 3, j + 1)$ can be applied simultaneously and before the rotations $G_{j+3,j+2}$ and $G_{j+2,j+1}$ for all $j = 1, 3, \ldots, m$ odd.

The use of Givens rotations to transform the matrix H_{k+2} into an upper triangular form is illustrated by the following example.

Example 4.1 Consider transforming the matrix $\bar{H}_5$ into an upper triangular form by using Givens rotations $\mathcal{G}_3 = G_{6,5}\cdots G_{2,1}G_{6,4}G_{4,2}$. Denoting a nonzero entry by x and a modified nonzero entry by $\tilde{\text{x}}$ (or by $\tilde{\tilde{\text{x}}}$ if the entry

$$\begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & x & x \end{bmatrix} \xrightarrow{G_{6,4}G_{4,2}} \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ 0 & \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} \\ 0 & 0 & \tilde{x} & \tilde{x} & \tilde{x} \\ 0 & 0 & 0 & \tilde{x} & \tilde{x} \\ 0 & 0 & 0 & 0 & \tilde{x} \end{bmatrix} \xrightarrow{G_{6,5}\cdots G_{2,1}} \begin{bmatrix} \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} \\ 0 & \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} \\ 0 & 0 & \tilde{\tilde{x}} & \tilde{\tilde{x}} & \tilde{\tilde{x}} \\ 0 & 0 & 0 & \tilde{\tilde{x}} & \tilde{\tilde{x}} \\ 0 & 0 & 0 & 0 & \tilde{\tilde{x}} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 4.1: Transforming $\bar{H}_5$ into an upper triangular matrix $\bar{R}_5$ by using plane rotations

has been modified twice), the process of transforming the matrix $\bar{H}_5$ into an upper triangular matrix $\bar{R}_5$ proceeds as presented in Figure 6.1.

Following Saad [105, Proposition 6.9, p. 169], we obtain the following proposition.

Proposition 4.1. *Let $\bar{R}_{m+2}$ and $\bar{q}_{m+2}$ be defined as in (4.6) and (4.7). In addition, let m be odd and $\Omega_j, j = 1, 3, \ldots, m$ be the rotation matrices as defined in (4.5) to transform $(\bar{I} + \bar{H}_{m+2})$ to upper triangular form. Denote by R_{m+2} the $(m+2)$-by-$(m+2)$ upper triangular matrix and by g_{m+2} the $(m+2)$-dimensional vector acquired after the last row and component of $\bar{R}_{m+2}$ and $\bar{g}_{m+2}$ have been deleted. Then we have the following properties:*

1. $\text{rank}((I+S)Q_{m+2}) = \text{rank}(R_{m+2})$
2. *Residual vector, which minimizes $||\beta e_1 - (\bar{I} + \bar{H}_{m+2})y_{m+2}||_2$ is given by*

$$y = R_{m+2}^{-1} g_{m+2} \tag{4.8}$$

3. *Residual at step m satisfies*

$$||b - (I+S)x_m||_2 = |\gamma_{m+3}| \tag{4.9}$$

Proof. 1. From (2.15) we obtain

$$\begin{aligned} (I+S)Q_{m+2} &= Q_{m+3}(\bar{I} + \bar{H}_{m+2}) \\ &= Q_{m+3}\mathcal{G}_m^*\mathcal{G}_m(\bar{I} + \bar{H}_{m+2}) \\ &= Q_{m+3}\mathcal{G}_m^*\bar{R}_{m+2} \end{aligned}$$

Since $Q_{m+3}\mathcal{G}_m^*$ is unitary, rank of $(I+S)Q_{m+2}$ equals rank of $\bar{R}_{m+2}$ which equals rank of R_{m+2}.

2. For any vector y_{m+2} it holds

$$\begin{aligned} ||\beta q_1 - (\bar{I} + \bar{H}_{m+2})y_{m+2}||_2^2 &= ||\mathcal{G}_m^* \mathcal{G}_m(\beta e_1 - (\bar{I} + \bar{H}_{m+2})y_{m+2})||_2^2 \\ &= ||\bar{g}_{m+2} - \bar{R}_{m+2}y_{m+2}||_2^2 \\ &= |\gamma_{m+3}| + ||g_{m+2} - R_{m+2}y_{m+2}||_2^2 \end{aligned}$$

The minimum is clearly reached when $||g_{m+2} - R_{m+2}y_{m+2}||_2^2$ is zero.

3. Since any vector in the basis generated by S can be written as $x_m = x_0 + Q_{m+2}y_{m+2}$, we have

$$\begin{aligned} b - (I + S)x_m &= b - (I + S)(x_0 + Q_{m+2}y_{m+2}) \\ &= r_0 - Q_{m+3}(\bar{I} + \bar{H}_{m+2})y_{m+2} \\ &= Q_{m+3}(\beta e_1 - (\bar{I} + \bar{H}_{m+2})y_{m+2}) \\ &= Q_{m+3}\mathcal{G}_m^* \mathcal{G}_m(\beta e_1 - (\bar{I} + \bar{H}_{m+2})y_{m+2}) \\ &= Q_{m+3}\mathcal{G}_m^*(\bar{g}_{m+2} - \bar{R}_{m+2}y_{m+2}) \end{aligned}$$

which may be minimized as in part (2). This leads to a relation

$$b - (I + S)x_m = Q_{m+3}\mathcal{G}_m^*(\gamma_{m+3}e_{m+3}) \tag{4.10}$$

where $Q_{m+3}\mathcal{G}_m^*$ is unitary and e_{m+3} denotes the $(m+3)$th column of an $(m+3)$-by-$(m+3)$ identity matrix. Since the 2-norm is unitarily invariant, we have $||Q_{m+3}\mathcal{G}_m^*(\gamma_{m+3}e_{m+3})||_2 = ||\gamma_{m+3}e_{m+3}||_2$ and the claim follows.

□

For the computational complexity of the two-sided Arnoldi-type process used in TSMRES to be equal to the standard Arnoldi process used by GMRES, we need a method to compute the columns h_{k+2} without performing an additional matrix-vector product and the related orthogonalizations. By a simple manipulation of the upper part of the relation (2.6), we can efficiently compute the columns h_{k+2} of $\bar{H}_{k+2}$, for $k = 1, 3, \ldots, m$ odd. This yields the following proposition.

Proposition 4.2. *Let $k = 1, 3, \ldots, m$ odd. Colums h_{k+2} of $\bar{H}_{k+2}$ can be computed as*

$$h_{k+2} = t_{k+2,k}^{-1}\left(\bar{e}_k - \bar{H}_{k+1}\bar{t}_k\right), \tag{4.11}$$

where $\bar{e}_k \in \mathbb{C}^{m+3}$ denotes the kth column of the matrix $\bar{I}$, $\bar{H}_{k+1}$ the top $k+3$ rows of the matrix $[h_1\ h_2\ \ldots\ h_{k+1}]$ and $\bar{t}_k$ the first $k+1$ components of t_k.

Proof. By the relation (2.6), we have $\hat{q}_{k+2} = S^{-1}q_k - \sum_{l=1}^{k+1}(S^{-1}q_k, q_l)q_l$, where q_j denotes the generated basis vectors of the two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$. After multiplication with S from the left, we obtain

$$S\hat{q}_{k+2} = q_k - \sum_{l=1}^{k+1} t_{l,k} S q_l, \tag{4.12}$$

where $t_{l,k} = (S^{-1}q_k, q_l)$ by the relation (2.8) and $\hat{q}_{k+2}$ denotes the $(k+2)$th basis vector before it has been scaled by $t^{-1}_{k+2,k} = ||q_{k+2}||_2$.

Multiplying (4.12) by $t^{-1}_{k+2,k}$ yields an expression for Sq_{k+2}. After multiplying from the left by Q^*_{k+3} and using orthogonality, for $h_{k+2} = Q^*_{k+3}Sq_{k+2}$ we then have

$$h_{k+2} = Q^*_{k+3} t^{-1}_{k+2,k} \left(q_k - \sum_{l=1}^{k+1} t_{l,k} S q_l \right) = t^{-1}_{k+2,k} \left(\bar{e}_k - \sum_{l=1}^{k+1} t_{l,k} h_l \right).$$

After writing the expression obtained for h_{k+2} in a matrix form, we have the claim. □

When the Givens rotations are used to solve the minimization problem, a practical difficulty in Algorithm 2.2 is the computation of the representation of S in $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ if the original matrix $\bar{H}_{k+1}$ is not stored. Then the relation (4.11) can not be directly used to compute the vectors h_{k+2}.

To compute the last column of $\bar{R}_{k+2}$ without having the matrix $\bar{H}_{k+1}$ directly available we have the following. For $k = 1, 3, \ldots, m$, after computing the basis vectors q_{k+2} and q_{k+3} we first apply the rotations from the previous iteration rounds to transform the matrix $\bar{I} + \bar{H}_k$ into an upper triangular form, i.e., we apply $\mathcal{G}_{k-2}$ to the vector $\bar{e}_{k+1} + h_{k+1}$. We then determine and apply the rotations $G_{k+3,k+1}$ and $G_{k+2,k+1}$ to clear the $(k+3)$th and $(k+2)$th entries of $e_{k+1} + h_{k+1}$ to have

$$\tilde{\mathcal{G}}_k(\bar{I} + \bar{H}_{k+1}),$$

where $\tilde{\mathcal{G}}_k = G_{k+2,k+1}G_{k+3,k+1}\mathcal{G}_{k-2}$ and $\bar{e}_k$, $\bar{H}_{k+1}$ and $\bar{t}_k$ defined as in Proposition 4.2.

By using the relations (4.11) and (4.6), for the vector $\bar{e}_{k+2} + h_{k+2}$ we then have

$$\tilde{\mathcal{G}}_k(\bar{e}_{k+2} + h_{k+2}) = \tilde{\mathcal{G}}_k \bar{e}_{k+2} + t^{-1}_{k+2,k} \tilde{\mathcal{G}}_k(\bar{e}_k - \bar{H}_{k+1}\bar{t}_k). \tag{4.13}$$

To compute the product $\tilde{\mathcal{G}}_k \bar{H}_{k+1}\bar{t}_k$, we use the relation (4.6). This yields

$$\tilde{\mathcal{G}}_k \bar{H}_{k+1}\bar{t}_k = \tilde{R}_{k+1}\bar{t}_k - \tilde{\mathcal{G}}_k \bar{t}_k,$$

where the matrix $\tilde{R}_{k+1}$ is of the size $(k+3)$-by-$(k+1)$ and contains the rotated columns of the matrix $\bar{H}_{k+1}$. From (4.13) it then follows

$$\hat{r}_{k+2} = \tilde{\mathcal{G}}_k(\bar{e}_{k+2} + \bar{h}_{k+2}) = \tilde{\mathcal{G}}_k\bar{e}_{k+2} + t_{k+2,k}^{-1}(\tilde{\mathcal{G}}_k(\bar{e}_k + \bar{t}_k) - \bar{R}_{k+1}\bar{t}_k).$$

From the vector $\hat{r}_{k+2}$ we then determine the Givens rotation $G_{k+3,k+2}$ to clear its $(k+3)$th entry. After applying $G_{k+3,k+2}$ we have $\bar{r}_{k+2}$, i.e., the $(k+2)$th column of the matrix $\bar{R}_{k+2}$, as

$$\bar{r}_{k+2} = G_{m+3,m+2}\hat{r}_{k+2}. \tag{4.14}$$

The relation (4.14) enables us to compute the matrix $\bar{R}_{k+2}$ directly without storing the original matrix $\bar{H}_{k+1}$.

In the following section, we describe a numerically stable implementation of TSMRES.

4.4 Numerically stabilized TSMRES

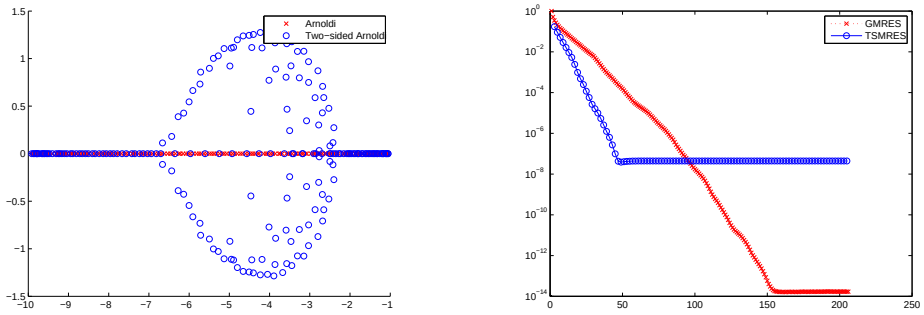
In Chapter 2 the actual numerical implementation of the TSMRES method was not considered. For motivation, we first consider the following numerical example illustrating numerical difficulties of Algorithm 2.2.

Example 4.2 Consider solving the linear system (2.1) where the matrix S is diagonal matrix having randomly generated uniformly distributed diagonal entries in the interval $(-10,-1)$, i.e., $\sigma(S) = \{x \mid x \in (-10,-1)\}$. We construct the spectrum of the matrix S by setting $\sigma(S) = U_n(-10,-1)$, where $n = 400$ and $U_n(a,b)$ is defined by the relation (2.34). The right-hand side is chosen as $b = \mathbf{1}_n/||\mathbf{1}_n||_2$, where $\mathbf{1}_n$ denotes a vector of all ones of length n.

The matrix S is well-conditioned. We have $||S||_2 \leq 10$, $||S^{-1}||_2 \leq 1$. Thus, for the condition number of the matrix S, we have the upper bound $\kappa(S) \leq 10$. Then by the numerical error result of the matrix-vector multiplication [61, pp. 69], the numerical error in the products Sq_{k+1} can be expected to be small. Also, since the matrix S is diagonal, the solutions computed to the linear systems $Sx = q_k$ only have a small componentwise error, see [61].

We let the dimension of the subspaces $\mathcal{K}_k(S;b)$ with $k = 2m+1$ and $\mathcal{K}^{\pm}_{\lceil\frac{m}{2}\rceil}(S;b)$ to equal 205 at most. To ensure orthogonality, the basis vectors of the two-sided subspace $\mathcal{K}^{\pm}_{\lceil\frac{m}{2}\rceil}(S;q_1)$ were computed with Algorithm 3.1 by using 2 orthogonalization steps at most. For both algorithms, we compute the residual norms from scratch each iteration round, i.e., no estimation of

the form (4.9) was used. Figure 4.2 shows the spectrum of S approximated with the standard Arnoldi and two-sided Arnoldi methods with subspaces of size 205 and the residual norms of the Householder GMRES [128] and TSMRES, i.e., Algorithm 2.2 without Proposition 4.2.



(a) Computed spectrum (b) Residual norm

Figure 4.2: Approximated spectrum and convergence with numerically unstable TSMRES

The Householder GMRES method is numerically backward stable [34]. Thus the solution computed with the Householder GMRES method can be expected to be accurate in a numerical sense when the matrix S is well-conditioned. By the results of Figure 4.2, once the size of the subspace $\mathcal{K}_k(S; q_1)$ has reached 205, the residual norm of GMRES has converged near the machine precision.

The numerical behaviour of TSMRES in this example is intriguing. In the following, we denote by Q_{m+2} the matrix having orthonormal columns spanning the two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$. Also let the entries of the $(m+3)$-by-$(m+2)$ matrix $\bar{H}_{m+2}$ be defined by the relations (2.9) and (2.11). Also denote by H_{m+2} the $(m+2)$-by-$(m+2)$ the matrix obtained after the $(m+3)$th row of $\bar{H}_{m+2}$ deleted. We note that since the dimension of the two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; b)$ equals 205 at most, the maximum degree of the polynomials π and φ in S and S^{-1} equals 102.

From Figure 4.2 we immediately observe that the eigenvalues of H_{m+2} approximating $\sigma(S)$ in the two-sided Krylov subspace are inaccurate. Even though $\sigma(S)$ is real, the approximations computed from H_{k+2} contain some approximate eigenvalues with a large imaginary part. This numerical behaviour of the computed eigenvalues seems to have a close resemblance to the effect of rounding errors in computing the roots of the Wilkinson polynomial of high degree, see [131, 133, 134].

The convergence behaviour of TSMRES is also affected. Initially TSMRES converges, but after the size of the subspace has reached the dimension of approximately 50, no reduction of the residual norm is achieved.

The basis vectors generated for the two-sided Krylov subspace are numerically nearly orthonormal, i.e., the computed quantity $||I_{k+2} - Q_{k+2}^* Q_{k+2}||_F$ is near machine precision. Thus the numerical error is contributing to the entries of the matrix H_{k+2}, i.e., the computed representation of the matrix S in the basis generated for $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$.

The generated set of basis vectors is numerically nearly orthogonal. The matrix $I + S$ is quite well-conditioned with a computed condition number estimate $\kappa^{\text{est}}(I + S) = 1.3E + 4$. Therefore we now consider the quantity $E = H_{m+2} - Q_{m+2}^* S Q_{m+2}$, which can be interpreted as the error between the computed and actual projections of the matrix S in the basis generated for the two-sided Krylov subspace $\mathcal{K}_{\lceil \frac{m}{2} \rceil}(S; b)$. Figure 4.3 shows the entries $e_{i,j}$ of E for which it holds $|e_{i,j}| \geq 1E - 14$.

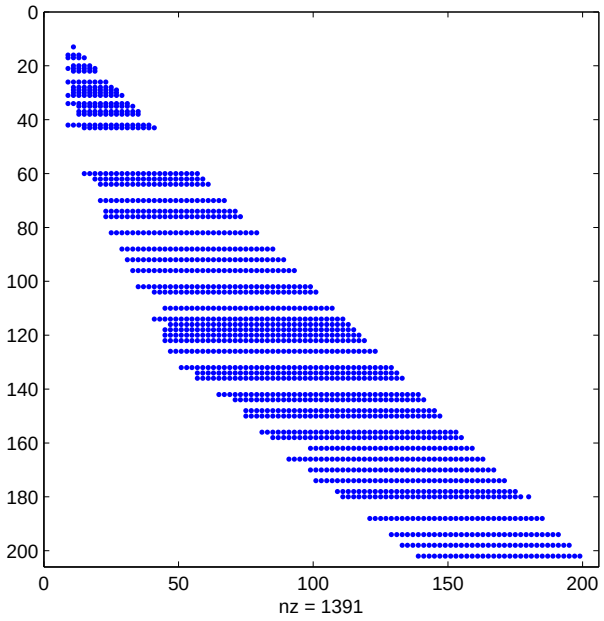


Figure 4.3: Entries $e_{i,j}$ of matrix $E = H_{205} - Q_{205}^* S Q_{205}$ for which $|e_{i,j}| \geq 1E - 14$

From Figure 4.3 we observe that H_{k+2} computed with Algorithm 2.1 is numerically not near the actual projection $Q_{m+2}^* S Q_{m+2}$. In fact, some entries in the lower triangular part of the matrix E are nearly one in magnitude. We

conclude, that in this particular example, a numerically accurate representation of the vectors Sq_{k+2} in the basis generated for the two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ cannot be computed with Algorithm 2.1 .

By Propositions 2.1 and 2.3, the two-sided Krylov subspaces $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ can be generated with Algorithm 2.1 in exact arithmetic. As show by Example 4.2, in floating point arithmetic Algorithm 2.1 may fail to construct a numerically accurate representation of the vectors Sq_{k+2} in the basis generated for the two-sided Krylov subspace.

From Example 4.2, we observe that since the computed quantity $||I - Q^*_{m+3}Q_{m+3}||_2$ is near machine precision, the basis vectors generated for the two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ are numerically nearly orthogonal. In spite of the near orthogonality, a representation of the vectors Sq_{k+2}, for $k = 1, 3, \ldots, m$ odd, in the basis generated cannot be computed in a numerically stable way for large k. Roughly speaking, this happens because *numerically*, due to the accumulating rounding errors, the basis vectors do not exactly form a basis for the two-sided subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ and Lemma 2.5 cannot be applied. A brief analysis of this phenomenon is given in Section 4.5.

In what follows, we present a numerically stable version of the TSMRES method. We modify TSMRES based on the observation that even though Algorithm 2.1 is unable to produce a representation of the vectors Sq_{k+2} in the two-sided Krylov subspace in a numerically stable way, the computed basis vectors are nearly orthonormal and form a basis for $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ in a numerical sense.

Denote by $Q_{m+2} = [q_1 \ q_2 \ \cdots \ q_{m+2}]$ the matrix having the generated basis vectors of $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ as its columns. In addition, denote by $Q^-_m = [q_1 \ q_3 \ q_5 \ \cdots \ q_m]$ and $Q^+_{m+1} = [q_1 \ q_2 \ q_4 \ \cdots \ q_{m+1}]$ the generated basis vectors associated with the operators S^{-1} and S, respectively. By the relation (2.10), for m odd, Algorithm 2.1 generates

$$\begin{aligned} S^{-1}Q^-_m &= Q_{m+3}\bar{T}_{m+1}, \\ SQ^+_{m+1} &= Q_{m+3}\bar{H}_{m+2}, \end{aligned}$$

with $\bar{T}_{m+1} = [t_1 \ t_3 \ \cdots \ t_m] \in \mathbb{C}^{(m+3)\times\lceil \frac{m}{2} \rceil}$ and $\bar{H}_{m+2} = [h_1 \ h_2 \ h_4 \ \cdots \ h_{m+1}] \in \mathbb{C}^{(m+3)\times(\lceil \frac{m}{2} \rceil+1)}$ where t_j and h_{j+1} are defined by the relations (2.8) and (2.9).

Let $q_1 = r_0/||r_0||_2$ with the initial residual $r_0 = b - (I + S)x_0$. We now consider solving the TSMRES minimization problem (2.20), defined as

$$\min_{v \in \mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)} ||b - (I + S)(x_0 + v)||_2.$$

Setting $v = v^+ + v^-$, we rewrite the TSMRES minimization problem to involve both S and S^{-1}. We have

$$\begin{aligned}
&\min_{v \in \mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S;q_1)} ||b - (I+S)(x_0+v)||_2 \\
&= \min_{(v^+ + v^-) \in \mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S;q_1)} ||r_0 - (I+S)v^+ - (I+S)v^-||_2, \\
&= \min_{\substack{v^+ \in \mathcal{K}_{\lceil \frac{m}{2} \rceil}(S;q_1), \\ \bar{v}^- \in \mathcal{K}_{\lceil \frac{m}{2} \rceil}(S^{-1};q_1)}} ||r_0 - (I+S)v^+ - (I+S^{-1})\bar{v}^-||_2, \qquad (4.15)
\end{aligned}$$

after setting $\bar{v}^- = Sv^-$. In the following, we describe how the acquired two-sided minimization problem (4.15) can be solved with the two-sided Krylov subspaces $\mathcal{K}_{\lceil \frac{m}{2} \rceil}(S;q_1)$ constructed with Algorithm 2.1.

In the basis generated for $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S;q_1)$, with m odd, we have

$$\begin{aligned}
(I^- + S^{-1})Q^-_m &= Q_{m+3}(\bar{I}^- + \bar{T}_{m+1}), \\
(I^+ + S)Q^+_{m+1} &= Q_{m+3}(\bar{I}^+ + \bar{H}_{m+2}),
\end{aligned} \qquad (4.16)$$

where $I^- = [e_1 \ e_3 \ \cdots \ e_m]$ and $I^+ = [e_1 \ e_2 \ \cdots \ e_{m+1}]$, e_j denotes the jth column of an n-by-n identity matrix and matrices $\bar{I}^-$ and $\bar{I}^+$ denote the first $(m+3)$ rows of I^- and I^+.

With (4.16), we now have the minimization problem (4.15) as

$$\begin{aligned}
&\min_{\substack{v^+ \in \mathcal{K}_{\lceil \frac{m}{2} \rceil}(S;q_1), \\ \bar{v}^- \in \mathcal{K}_{\lceil \frac{m}{2} \rceil}(S^{-1};q_1)}} ||r_0 - (I+S)v^+ - (I+S^{-1})\bar{v}^-||_2 \\
&= \min_{\substack{y^+_m \in \mathbb{C}^{\lceil \frac{m}{2} \rceil + 1}, \\ \bar{y}^-_m \in \mathbb{C}^{\lceil \frac{m}{2} \rceil}}} ||r_0 - (I^+ + S)Q^+_{m+1}y^+_m - (I^- + S^{-1})Q^-_m\bar{y}^-_m||_2 \\
&= \min_{\substack{y^+_m \in \mathbb{C}^{\lceil \frac{m}{2} \rceil + 1}, \\ \bar{y}^-_m \in \mathbb{C}^{\lceil \frac{m}{2} \rceil}}} ||\beta e_1 - (\bar{I}^+ + \bar{H}_{m+2})y^+_m - (\bar{I}^- + \bar{T}_{m+1})\bar{y}^-_m||_2. \qquad (4.17)
\end{aligned}$$

This is equivalent to the original formulation (2.20) in the two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S;q_1)$. Once the minimizers y^+_m and $\bar{y}^-_m$ have been computed, it remains to set $v^+ = Q^+_{k+2}y^+_m$, $\bar{v}^- = Q^-_{k+1}\bar{y}^-_m$ and compute $v^- = S^{-1}\bar{v}^-$ to have the approximate solution as $x_m = x_0 + z_m = x_0 + (v^+ + v^-)$.

To find the minimizers y^+_m and $\bar{y}^-_m$, we need to solve a $(m+3)$-by-$(m+2)$ least-squares problem. Writing the acquired minimization problem (4.17) in a matrix form yields

$$\min_{\substack{y^+_m \in \mathbb{C}^{\lceil \frac{m}{2} \rceil + 1}, \\ \bar{y}^-_m \in \mathbb{C}^{\lceil \frac{m}{2} \rceil}}} ||\beta e_1 - [(\bar{I}^+ + \bar{H}_{m+2}) \ (\bar{I}^- + \bar{T}_{m+1})] \begin{bmatrix} y^+_m \\ \bar{y}^-_m \end{bmatrix} ||_2.$$

Denote by e_j the jth column of a $(m+2)$-by-$(m+2)$ identity matrix. Then, after applying a permutation

$$P = [e_1 \; e_{\lceil \frac{m}{2} \rceil + 2} \; e_2 \; e_{\lceil \frac{m}{2} \rceil + 3} \; e_3 \; \cdots \; e_{m+2} \; e_{\lceil \frac{m}{2} \rceil + 1}],$$

for the least-squares approach we have

$$\bar{Y}_{m+2} y = \beta e_1, \tag{4.18}$$

where $\bar{Y}_{m+2}$ denotes a $(m+3)$-by-$(m+2)$ upper Hessenberg matrix, defined as

$$\bar{Y}_{m+2} = [(e_1 + h_1) \; (e_1 + t_1) \; (e_2 + h_2) \; \cdots \; (e_m + t_m) \; (e_{m+1} + h_{m+1})] \tag{4.19}$$

and

$$y = P \begin{bmatrix} y_m^+ \\ \bar{y}_m^- \end{bmatrix}.$$

Once the least-squares problem (4.18) has been solved, it remains to set $P^+ = [e_1 \; e_3 \; \cdots \; e_{m+2}]$ and $P^- = [e_2 \; e_4 \; \cdots \; e_{m+1}]$ to have the computed minimizers as $y_m^+ = P^+ y$ and $\bar{y}_m^- = P^- y$.

Let ORTHOGON denote an algorithm which orthonormalizes a given vector against a given orthonormal basis, defined as in Algorithm 3.1. For the alternate formulation of TSMRES, we have Algorithm 4.1.

Algorithm 4.1 Numerically stabilized TSMRES

1: Compute $r_0 = b - x_0 - Sx_0$, $\beta = ||r_0||_2$, $q_1 = r_0/\beta$
2: $w = Sq_1$, $h_{1,1} = (w, q_1)$, $w = w - h_{1,1}q_1$, $h_{2,1} = ||w||_2$, $q_2 = w/h_{2,1}$
3: **for** $j = 1, 2, \ldots, \frac{m+1}{2}$ **do**
4: $\quad w = S^{-1}q_{2j-1}$
5: $\quad [q_{2j+1}, [t_{1,2j-1}, \ldots, t_{2j+1,2j-1}]]$ =ORTHOGON$([q_1, \ldots, q_{2j}], w)$
6: $\quad w = Sq_{2j}$
7: $\quad [q_{2j+2}, [h_{1,2j}, \ldots, h_{2j+2,2j}]]$ =ORTHOGON$([q_1, \ldots, q_{2j+1}], w)$
8: **end for**
9: Compute $y = \text{Argmin}_y ||\beta e_1 - \bar{Y}_{m+2} y||_2$
10: Set $y_m^+ = P^+ y$ and $\bar{y}_m^- = P^- y$
11: Compute $x_m = x_0 + Q_{m+2}^+ y_m^+ + S^{-1} Q_{m+1}^- \bar{y}_m^-$

As before, in the practical implementation of Algorithm 4.1, we solve the least-squares problem (4.18) by transforming the upper Hessenberg matrix $\bar{Y}_{m+2}$ into an upper triangular form by using plane rotations. The main difference to efficiently solving the original TSMRES minimization problem (2.20) is, that for the jth iteration round with $j = 1, 3, \ldots, m$ odd, the plane

rotations have only to be applied to clear the entries at positions $(j+2, j+1)$ and $(j+3, j+2)$.

Denote by $G_{j,k}$ the Givens rotation to clear element at position (j,k). For the jth iteration round with $j = 1, 3, \ldots, m$ odd, set $\hat{\Omega}_j = G_{j+3,j+2}G_{j+2,j+1}$ to clear entries at positions $(j+2, j+1)$ and $(j+3, j+2)$. Denoting by $\mathcal{G}_m$ the product of these rotations, we have

$$\mathcal{G}_m = \hat{\Omega}_m \hat{\Omega}_{m-2} \cdots G_{2,1}. \tag{4.20}$$

With $\mathcal{G}_m$, similarly to (4.6) and (4.7), we then have

$$\bar{R}_{m+2} = \mathcal{G}_m \bar{Y}_{m+2}, \tag{4.21}$$

$$\bar{g}_{m+2} = \mathcal{G}_m(\beta e_1) = (\gamma_1, \ldots, \gamma_{m+3})^T, \tag{4.22}$$

where $\bar{R}_{m+2}$ again denotes an upper triangular $(m+3)$-by-$(m+2)$ matrix with the last row containing only zeroes. Similarly to Proposition 4.1, we have the following.

Proposition 4.3. *Let $\bar{R}_{m+2}$ and $\bar{q}_{m+2}$ be defined as in (4.21) and (4.22). In addition, let m be odd and $\hat{\Omega}_j, j = 1, 3, \ldots, m$ be the rotation matrices as defined in (4.20) to transform $\bar{Y}_{m+2}$ to upper triangular form. Denote by R_{m+2} the $(m+2)$-by-$(m+2)$ upper triangular matrix and by g_{m+2} the $(m+2)$-dimensional vector acquired after the last row and component of $\bar{R}_{m+2}$ and $\bar{g}_{m+2}$ have been deleted. Then we have the following properties:*

1. $\operatorname{rank}(\bar{Y}_{m+2}) = \operatorname{rank}(R_{m+2})$

2. *Residual vector, which minimizes $||\beta e_1 - \bar{Y}_{m+2} y||_2$ is given by*

$$y = R_{m+2}^{-1} g_{m+2} \tag{4.23}$$

3. *Residual at step m satisfies*

$$||b - (I+S)x_m||_2 = |\gamma_{m+3}| \tag{4.24}$$

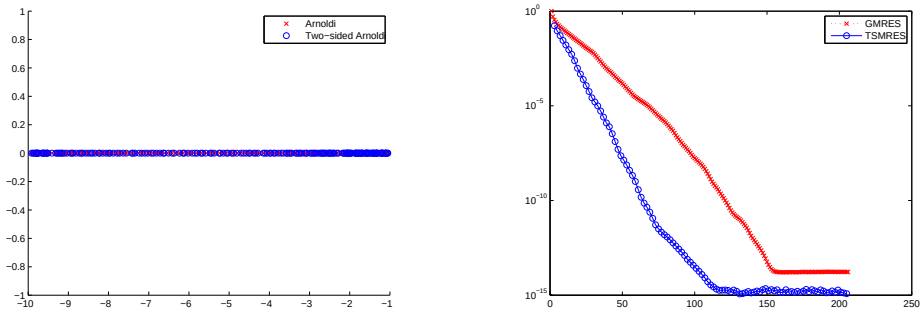
Proof. The steps of the proof exactly follow to those of Proposition 4.1, but with alternate definitions of R_{m+2} and g_{m+2} and the minimization problem (4.15). □

We now reconsider Example 4.2 to illustrate the stability of the numerically stabilized TSMRES method.

Example 4.3 Let the matrix S and the vector b be equal to those used in Example 4.2. As previously, we let the dimension of the subspaces $\mathcal{K}_k(S;b)$ with $k = 2m+1$ and $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S;b)$ to equal 205 at most.

After an orthonormal basis of the two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S;q_1)$ has been generated, we compute the projections $\hat{h}_{k+2} = Q^*_{m+3} S q_{k+2}$ for $k = 1, 3, \ldots, m$ and set $\hat{H}_{m+2} = \begin{bmatrix} h_1 & h_2 & \hat{h}_3 & \cdots & h_{m+1} & \hat{h}_{m+2} \end{bmatrix}$. With the two-sided Arnoldi method, we then approximate the eigenvalues of the matrix S in the basis generated for $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S;q_1)$ by computing the eigenvalues of the matrix $\hat{H}_{m+2}$. Figure 4.3 shows the spectrum of S approximated with the standard Arnoldi method and with the two-sided Arnoldi method using the computed $\hat{H}_{m+2}$ for a subspace dimension 205.

Figure 4.3 also describes the residual norms of the Householder GMRES and numerically stabilized TSMRES methods. As previously, the residual norms have been computed from scratch each iteration round.



(a) Computed spectrum (b) Residual norm

Figure 4.4: Approximated spectrum and convergence of numerically stabilized TSMRES for Example 4.2

For the two-sided subspaces, the approximate eigenvalues computed from the matrix $\hat{H}_{m+2}$ seem numerically much more accurate than the ones approximated from the matrix $\bar{H}_{m+2}$ computed with the two-sided Arnoldi method, i.e., Algorithm 2.1. The residual norm of the numerically stabilized TSMRES is also improved when compared to the residual norm of the standard TSMRES method. As the size of the subspace increases, the residual is reduced as is expected. The numerically stabilized TSMRES method is able to reduce the residual norm approximately to the same level as the Householder GMRES method.

From Figure 4.3 we also observe, that after the dimension of the two-sided Krylov subspace has reached approximately 50, the speed of which

the residual norm reduces per iteration step of TSMRES is reduced slightly. The reduction is due to the least squares problem (4.17) of the numerically stabilized TSMRES becoming nearly rank deficient.

In our numerical experiments with Algorithm 2.2, with and without Proposition 4.2, we observed behaviour similar to Example 4.2 in several other cases. In those cases, the numerical instability was observed when the dimension of the two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ was large, usually well over 50. On the other hand, we never encounter a case where the numerically stabilized TSMRES, i.e., Algorithm 4.1 was unstable.

In the following section, we briefly consider the numerical stability of the different formulations of TSMRES.

4.5 Aspects of the numerical stability of TSMRES

It is well known that the GMRES method is numerically backward stable. The Householder implementation of GMRES, proposed by Walker in [128], has been shown to be backward stable in the normwise sense in [34]. In the Householder implementation of GMRES, the Householder transformations are used to construct an orthonormal basis of the Krylov subspace $\mathcal{K}_m(A; q_1)$. In [94] Paige, Rozložník and Strakoš proved that under some mild assumptions also the modified Gram-Schmidt version of GMRES is backward stable.

In this section, we consider the effect of rounding errors in the computation of the two-sided subspace $\mathcal{K}^{\pm}_j(S; q_1)$ and the computation of the representation of the matrix S in the basis generated. We do not claim to establish the backward stability of the numerically stabilized TSMRES method, however. The aim is to present arguments supporting the fact that the stabilized formulation seems to be numerically more stable in practical computations. As is customary in the floating point error analysis, throughout this section we assume real arithmetic.

In the following, we denote by $\tilde{\gamma}_k$ the quantity

$$\tilde{\gamma}_k = \frac{cku}{1 - cku}, \tag{4.25}$$

where u is the machine precision and c is a small integer constant whose exact value is unimportant. In this section, we use "hats" to refer to the computed quantities and denote by $fl(\cdot)$ the floating point result of the operation $(\cdot)$.

We now consider the numerical error arising in the computation of a two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$. We begin by observing that both

implementations of the TSMRES method compute a QR-decomposition of the matrix

$$[q_1 \; Sq_1 \; S^{-1}q_1 \; Sq_2 \; \cdots \; S^{-1}q_m], \tag{4.26}$$

where q_j denotes the jth orthonormal basis vector of $\mathcal{K}_{\lceil \frac{m}{2} \rceil}(S; q_1)$. Let $\mathcal{K} = [Sq_1 \; S^{-1}q_1 \; \cdots \; S^{-1}q_m]$. Then the matrix $[q_1 \; \mathcal{K}]$ is of the size n-by-$(m+3)$. By the relation (2.10), we then have the QR-decomposition of the matrix $[q_1 \; \mathcal{K}]$ as

$$[q_1 \; \mathcal{K}] = [q_1 \; q_2 \; \cdots \; q_{m+3}][\beta e_1 \; h_1 \; t_1 \; h_2 \; t_3 \; h_4 \; \cdots \; t_m \; h_{m+1}] = Q_{m+3}R_{m+3}, \tag{4.27}$$

where $R_{m+3} \in \mathbb{R}^{(m+3)\times(m+3)}$ is an upper triangular matrix and the matrix $Q_{m+3} \in \mathbb{R}^{n\times(m+3)}$ has the generated basis vectors of the two-sided Krylov subspace $\mathcal{K}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ as its columns. By rewriting the relation (4.27) for the two-sided Arnoldi process presented in Section 2.3, we have

$$\mathcal{K} = Q_{m+3}\bar{H}_{m+2}, \tag{4.28}$$

where $\bar{H}_{m+2} = [h_1 \; t_1 \; h_2 \; \cdots \; t_m \; h_{m+1}]$ is a $(m+3)$-by-$(m+2)$ matrix with a Hessenberg structure.

For the standard Arnoldi process, the basis generated with the Householder transformations is known to be numerically nearly orthonormal independent of the matrix A [133]. A similar result was conjectured to hold also for the iterated modified Gram-Schmidt process by Hoffmann in [64]. When the matrix A is of full rank, such a result was shown to hold with two orthogonalization steps by Giraud et al. in [53].

To our knowledge no rounding error results concerning the two-sided Arnoldi process exist. Let κ be positive and chosen as in Algorithm 3.1. Similarly to [34], see also [64], by the results of several numerical experiments we believe that it is plausible to assume the near orthonormality of the basis computed with Algorithm 4.1.

Assumption 4.4. *For sufficiently small κ, the basis generated with Algorithm 4.1 satisfies*

$$||I - \hat{Q}_{m+3}^T\hat{Q}_{m+3}||_F \leq \zeta\kappa(m+3)^\alpha u \tag{4.29}$$

while the total number of reorthogonalization steps does not exceed $\xi(m+3)$. Here ξ and α are constants close to one and ζ is a constant independent of $(m+3)$, κ and u.

If the condition (4.29) holds, it then follows that for the computed $\hat{Q}_{m+3}$ there exists an exactly orthonormal matrix $\bar{Q}_{m+3}$ such that $\hat{Q}_{m+3} = \bar{Q}_{m+3} + \Delta Q_{m+3}$ and the condition

$$||\Delta Q_{m+3}||_2 = ||\hat{Q}_{m+3} - \bar{Q}_{m+3}||_F \leq \zeta\kappa(m+3)^\alpha u,$$

holds, i.e., the computed set of basis vectors for $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ is numerically very close to being exactly orthonormal.

For a basis which is exactly orthonormal, in finite precision arithmetic we have

$$fl(fl(\hat{\mathcal{K}}) + \Delta\mathcal{K}) = \bar{Q}_{m+3}\hat{H}_{m+2}, \tag{4.30}$$

with $\Delta\mathcal{K}$ arising from the finite precision orthogonalization of the computed $fl(\hat{\mathcal{K}}) = \hat{\mathcal{K}} + \Delta\mathcal{K}_S$. The error term $\Delta\mathcal{K}_S$ in $fl(\hat{\mathcal{K}})$ is of the form

$$\Delta\mathcal{K}_S = [\Delta S\hat{q}_1 \ \Delta S^{-1}\hat{q}_2 \ \cdots \ \Delta S^{-1}\hat{q}_m], \tag{4.31}$$

where $\Delta S\hat{q}_j$ and $\Delta S^{-1}\hat{q}_j$ denote the errors from the finite precision computation of the matrix vector product $z_j = S\hat{q}_j$ and solution of the linear system $Sz_j = \hat{q}_j$.

Since the two-sided Arnoldi process constructs a QR-decomposition of the computed matrix $[\hat{q}_1 \ \hat{\mathcal{K}}]$, the standard floating point error analysis can be used. The following theorem is from Higham [61, pp. 372].

Theorem 4.5. *Suppose the modified Gram-Schmidt method is applied to $A \in \mathbb{R}^{m\times n}$ of rank n, yielding computed matrices $\hat{Q} \in \mathbb{R}^{m\times n}$ and $\hat{R} \in \mathbb{R}^{n\times n}$. Then there are constants $c_i \equiv c_i(m,n)$ such that*

$$A + \Delta A_1 = \hat{Q}\hat{R}, \quad ||\Delta A_1||_2 \le c_1 u||A||_2, \tag{4.32}$$
$$||\hat{Q}^T\hat{Q} - I||_2 \le c_2 u\kappa(A) + O((u\kappa(A))^2), \tag{4.33}$$

and there exists an orthonormal matrix $\bar{Q}$ such that

$$A + \Delta A_2 = \bar{Q}\hat{R}, \quad ||\Delta A_{2,j}||_2 \le c_3 u||a_j||_2, \quad j = 1,\ldots,n \tag{4.34}$$

where $\Delta A_{2,j}$ denotes the jth column of ΔA_2.

By Theorem 4.5 and the relation (4.30) it then follows, with an exactly orthonormal matrix $\bar{Q}_{m+3}$,

$$\bar{Q}_{m+3}(fl(\hat{\mathcal{K}}) + \Delta\mathcal{K}) = \hat{H}_{m+2},$$

with the error bound $||\Delta\mathcal{K}||_2 \le c_1 u||\mathcal{K} + \Delta\mathcal{K}_S||_2$. Since the matrix $\bar{Q}_{m+3}$ is exactly orthonormal, we then have

$$\hat{\mathcal{K}} + E = \bar{Q}^T_{m+3}\hat{H}_{m+2}, \tag{4.35}$$

where $||E||_2 \le ||\Delta\mathcal{K}_S||_2 + c_1 u(||\mathcal{K}||_2 + ||\Delta\mathcal{K}_S||_2)$

We now consider the error term E in more detail. For $\hat{\mathcal{K}}$, we have

$$||\hat{\mathcal{K}}||_2 = |||[S\hat{q}_1 \ S^{-1}\hat{q}_1 \ \cdots \ S^{-1}\hat{q}_m]||_2 = |||[S\hat{Q}^+ \ S^{-1}\hat{Q}^-]P||_2, \tag{4.36}$$

where P denotes permutation to rearrange the columns of $\hat{\mathcal{K}}$ and $\hat{q}_j$ denotes the jth column of the matrix $\hat{Q}_{m+3}$. Analogously as in Section 4.4, the matrices $\hat{Q}^+$ and $\hat{Q}^-$ contain the computed basis vectors related to the operators S and S^{-1} such that $\hat{Q}^+ = [\hat{q}_1\ \hat{q}_2\ \hat{q}_4\ \cdots\ \hat{q}_{m+1}]$ and $\hat{Q}^- = [\hat{q}_1, \hat{q}_3\ \hat{q}_5\ \ldots\ \hat{q}_m]$. In the following, we also use similar definitions for the exactly orthonormal matrices $\bar{Q}^+$ and $\bar{Q}^-$ and the error terms ΔQ^+ and ΔQ^-.

By the properties of the operator norm we have

$$\begin{aligned} |||[S\hat{Q}^+\ S^{-1}\hat{Q}^-]P||_2 &= |||[S(\bar{Q}^+ + \Delta Q^+)\ S^{-1}(\bar{Q}^- + \Delta Q^-])||_2 \\ &\leq |||[S\ S^{-1}]|||_2||\bar{Q}^{\pm} + \Delta Q^{\pm}||_2 \\ &\leq |||[S\ S^{-1}]|||_2\big(||\bar{Q}^{\pm}||_2 + ||\Delta Q^{\pm}||_2\big), \end{aligned}$$

where $\bar{Q}^{\pm}$ and $\Delta Q^{\pm}$ are $2n$-by-$(m+2)$ matrices defined as

$$\bar{Q}^{\pm} = \begin{bmatrix} \bar{Q}^+ & \mathbf{0} \\ \mathbf{0} & \bar{Q}^- \end{bmatrix}, \quad \Delta Q^{\pm} = \begin{bmatrix} \Delta Q^+ & \mathbf{0} \\ \mathbf{0} & \Delta Q^- \end{bmatrix}.$$

For $|||[S\ S^{-1}]|||_2$ we have

$$\begin{aligned} |||[S, S^{-1}]^T||_2 &= \max_{||x||_2=1} |||[S, S^-]^T x||_2 \\ &= \max_{||x||_2=1} (||S^T x||_2^2 + ||S^{-T} x||_2^2)^{1/2} \\ &\leq (||S^T||_2^2 + ||S^{-T}||_2^2)^{1/2} \\ &\leq \sqrt{2}\max\{||S||_2, ||S^{-1}||_2\}. \end{aligned}$$

The matrix $\bar{Q}^{\pm}$ has exactly orthonormal columns, i.e., $||\bar{Q}^{\pm}||_2 = 1$. Since the columns of ΔQ^+ and ΔQ^+ are extracted from the columns of ΔQ_{m+3}, it follows that $||\Delta Q_{m+3}||_2 \geq \max\{||\Delta Q^+||_2, ||\Delta Q^-||_2\}$. Then by Assumption 4.4 we have for $\Delta Q^{\pm}$ the inequality

$$||\Delta Q^{\pm}||_2 = \max\{||\Delta Q^+||_2, ||\Delta Q^-||_2\} \leq \zeta\kappa(m+3)^{\alpha}u.$$

For $\hat{\mathcal{K}}$ we then have

$$||\hat{\mathcal{K}}||_2 \leq \sqrt{2}\max\{||S||_2, ||S^{-1}||_2\}\big(1 + \zeta\kappa(m+3)^{\alpha}u\big). \tag{4.37}$$

A similar analysis for the term $\Delta\mathcal{K}_S$ yields

$$||\Delta\mathcal{K}_S||_2 \leq \sqrt{2}\max\{||\Delta S||_2, ||\Delta S^{-1}||_2\}\big(1 + \zeta\kappa(m+3)^{\alpha}u\big),$$

with ΔS and ΔS^{-1} as before.

Denote by $|X|$ a matrix with elements $|x_{i,j}|$ and let the inequalities between such matrices hold componentwise. To bound the term ΔS, we use an elementary backward error result of matrix-vector multiplication, given for completeness in the following [61, pp. 69].

Lemma 4.6. *Let $A \in \mathbb{R}^{m\times n}$, $x \in \mathbb{R}^n$ and $y = Ax$. Then, for the computed matrix vector product $\hat{y}$, we have*

$$\hat{y} = (A + \Delta A)x, \quad |\Delta A| \leq \gamma_n |A|. \tag{4.38}$$

By the standard norm inequalities, equation (4.38) *then implies* $||\Delta A||_2 \leq \sqrt{\min\{m,n\}}\gamma_n ||A||_2$.

By using Lemma 4.6, for the term ΔS arising from the matrix vector products $S\hat{q}_{k+1}$ with $k = 1, \ldots, m$ we then have

$$||\Delta S||_2 \leq \sqrt{n}\gamma_n ||S||_2 + O(u^2). \tag{4.39}$$

We note that a better error bound could be obtained by taking into account the sparsity of S.

It remains to bound the term ΔS^{-1}, arising from the solution of the linear systems $Sz = \hat{q}_k$, $k = 1, \ldots, m$. For the computed solution $\hat{z}$ we have

$$(S + \Delta\hat{S}_{-1})\hat{z} = \hat{q}_k,$$

where $\Delta\hat{S}_{-1}$ is a bounded error term. Roughly we cannot expect the error ΔS^{-1} to be better than ΔS, since solving a linear system usually involves at least as many operations as matrix-vector multiplication.

When solving linear systems with the matrix S is not numerically stable, the term $\Delta\hat{S}_{-1}$ will be large and dominate the total error of the two-sided orthogonalization process. We now assume that solving linear systems with S is backward stable. For the term ΔS_{-1} it then follows

$$||\Delta S_{-1}||_2 = ||\hat{z} - z||_2 \leq C\kappa(S)u||z||_2,$$

where constant C is independent of S and $\hat{q}_k$. Since $\hat{q}_j$ is nearly of unit length, for the norm of the exact solution z it holds $||z||_2 = ||S^{-1}\hat{q}_k||_2 = ||S^{-1}||_2 + ||S^{-1}||_2||\Delta q_k||_2$, where Δq_k denotes the kth column of ΔQ_{m+3}. This yields the backward error as

$$||\Delta S_{-1}||_2 \leq Cu\kappa(S)||S^{-1}||_2 + O(u^2).$$

For the error term E, we then finally have the bound

$$\begin{aligned}
||E||_2 &\leq ||\Delta\mathcal{K}_S||_2 + c_1 u(||\mathcal{K}||_2 + ||\Delta\mathcal{K}_S||_2) \\
&\leq \sqrt{2}\max\{||\Delta S||_2, ||\Delta S^{-1}||_2\} && (4.40)\\
&\quad + c_1 u(\sqrt{2}\max\{||S||_2, ||S^{-1}||_2\}) + O(u^2) \\
&\leq \sqrt{2}\max\{\sqrt{n}\gamma_n||S||_2, Cu\kappa(S)||S^{-1}||_2\} && (4.41)\\
&\quad + c_1 u(\sqrt{2}\max\{||S||_2, ||S^{-1}||_2\}) + O(u^2), && (4.42)
\end{aligned}$$

i.e., the error depends on the norms $||S||_2$ and $||S^{-1}||_2$.

The bound (4.42) explains why Theorem 4.2 is not numerically stable for large values of k. When the vectors h_{k+2} are computed as

$$h_{k+2} = t_{k+2,k}^{-1} \left(\bar{e}_k - \bar{H}_{k+1} \bar{t}_k \right),$$

the computed equivalents of the terms $\bar{t}_k$ and $\bar{H}_{k+1}$ contain numerical error depending on the norms $||S||_2$ and $||S^{-1}||_2$. Thus the computed $\hat{h}_{k+2}$ will contain an error which is the product of the errors of the computed terms $\bar{H}_{k+1}$ and $\bar{t}_k$ in magnitude. Unless the error terms cancel out, repeatedly using the relation (4.11) will increase the numerical error exponentially and therefore cannot be expected to be numerically stable for large values of m.

We now consider Lemma 2.5 in finite precision arithmetic. Even if Q_{k+3} would be exactly orthogonal, neglecting the error from arising in the orthogonalization process, we have the computed $\hat{h}_{k+2}$ as

$$\begin{aligned} \hat{h}_{k+2} &= Q_{k+3}^T (S + \Delta S) q_{k+2} \\ &= Q_{k+3}^T (S + \Delta S)((S^{-1} + \Delta S_{-1}) q_k - Q_{k+1} \bar{t}_k)/t_{k+2,k} \\ &= h_{k+2} + \Delta h_{k+2}, \end{aligned}$$

where

$$||\Delta h_{k+2}||_2 \leq ||S^{-1} \Delta S||_2 + ||S \Delta S_{-1}||_2 + ||\Delta S||_2 ||\bar{t}_k||_2,$$

where ΔS and ΔS_{-1} are bounded as before, i.e., the numerical error is multiplied by the norms $||S||_2$ and $||S^{-1}||_2$, but the growth is not exponential.

Algorithm 4.1, i.e., the numerically stabilized TSMRES method avoids computing the columns h_{k+2} explicitly. Therefore only sources of the numerical error are the construction of an orthonormal basis for the two-sided Krylov subspace and the solution of $(m+3)$-by-$(m+2)$ least-squares problem. Therefore, in a numerical sense, the minimizers produced by the numerically stabilized TSMRES method can be expected to be nearly as good as they can be in the basis generated for the two-sided Krylov subspace. In spite of this, as shown by the latter part of Example 4.2, the least-squares problem obtained can be numerically nearly ill-conditioned even for well-conditioned matrices S and S^{-1}. Thus, at this point we do not attempt to make claims of backward stability.

We now analyze the computational cost of both versions of the TSMRES method.

4.6 Computational cost

In this section, we analyze the computational cost and storage requirements of TSMRES. We compare the computational cost of both implementations

of TSMRES and relate them to the computational cost of the GMRES and Householder GMRES methods. In both implementations of the TSMRES method, most of the floating point operations arise from the computation of matrix-vector products and orthogonalization of basis vectors. Since generally $m \ll n$, the computational cost of applying plane rotations to transform $\bar{H}_{k+2}$ to upper triangular form is of the order $O(m^2)$, which is neglible and therefore omitted.

We first consider the standard TSMRES method, i.e., Algorithm 2.2, which uses the two-sided Gram-Schmidt process to construct an orthonormal basis for the two-sided Krylove subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$. The construction of q_2 on line 2 requires one matrix-vector product and approximately $O(7n)$ operations for orthogonalization. The computation of q_{k+2} on lines from 4 to 8 requires a matrix-vector product and $k+1$ inner products and vector updates, requiring

$$S_1 = \sum_{l=1}^{k+1} \big(n + (n-1) + 2n\big) + 2n + n = (k+1)(4n-1) + 3n$$

operations for orthogonalizations in total. Similarly, to generate q_{k+3} on lines from 9 to 13 we need a matrix-vector product and

$$S_2 = \sum_{l=1}^{k+2} \big(n + (n-1) + 2n\big) + 2n + n = (k+2)(4n-1) + 3n$$

operations for orthogonalizations. Assuming that the line 16 is computed by using the relation (2.11), we need one matrix-vector product and $k+3$ inner products and vector updates, which yields

$$S_3 = \sum_{l=1}^{k+3} \big(n + (n-1) + 2n\big) = (k+3)(4n-1)$$

operations to compute h_{k+2} each round. Combining the parts S_1, S_2 and S_3, we then get the computational cost of one step of Algorithm 2.2. Since each step of the iteration increases the size of subspace by two, to generate a Krylov subspace of size $m+3$, we only need to take $\frac{1}{2}(m+1)$ steps in total. Setting $k = 2j-1$, the computational cost of generating an orthonormal basis for $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ becomes

$$\sum_{j=1}^{(m+1)/2} \big(S_1 + S_2\big) = O(2nm^2 + 13nm + 11n)$$

operations for the orthogonalizations and $(m+1)$ matrix-vector products in total. In addition, to compute a representation of the vectors Sq_k in the Krylov subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$, we need to compute h_{k+2}, for each $k = 1, 3, \ldots, m$ by using the relation (2.11). After setting $k = 2j-1$, we obtain that this requires

$$\sum_{j=1}^{(m+1)/2} S_3 = O(nm^2 + 8nm + 7n)$$

operations and $\frac{1}{2}(m+1)$ matrix-vector products in total.

We now consider the effect of using Proposition 4.2 to compute the vectors h_{k+2}. If the relation (4.11) is used, we do not need to compute any additional matrix vector products or orthogonalizations. With (4.11) the computation of the vectors h_{k+2} requires

$$\tilde{S}_3 = (k+3)\big((k+1)+k\big) + 1 + (k+3)$$

floating point operations at each step of the iteration. Since the vector h_{k+2} needs to be computed $\frac{1}{2}(m+1)$ times in total, after setting $k = 2j-1$ we again have the computational cost as

$$\sum_{j=1}^{(m+1)/2} \tilde{S}_3 = O(\frac{1}{3}m^3).$$

The cost is independent of n and is usually neglible since often $m \ll n$.

We now consider the computational complexity of the numerically stabilized TSMRES, i.e., Algorithm 4.1 which uses the modified Gram-Schmidt process with reorthogonalization to construct the two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$.

We begin by considering the computational cost of Algorithm 3.1, which orthonormalizes a given vector w against a given orthonormal basis $Q_k = \{q_1, q_2, \ldots, q_k\}$ and produces a vector q_{k+1} which is orthogonal projection to the complement of $\text{span}\{q_1, \ldots, q_k\}$.

Denote by N_r the number of orthogonalization steps required. Each inner step in the orthogonalization process requires k inner products and vector updates. Therefore, to compute a single inner step approximately

$$S_0 = \sum_{j=1}^{k} \big(n + (n-1) + 2n + 1\big) = 4kn$$

operations are required. Including the additional operations required for checking if the generated vector is numerically in the span of Q_k, to compute q_{k+1} with N_r orthogonalization steps we need

$$S_1 = 2n + N_r(2n + S_0 + 2n + 4 + 2) + n = N_r(4kn + 4n + 6) + 3n \quad (4.43)$$

operations in total. Thus, when only a single orthogonalization step is used, the cost is practically equivalent to the cost of the modified Gram-Schmidt process without reorthogonalization.

With the computational cost of Algorithm 3.1 determined, we now consider the cost of Algorithm 4.1. The computation of q_{k+2} on lines 4 and 5 requires one matrix-vector product and an orthogonalization against a basis of size $k+1$. Therefore, by (4.43)

$$S_1 = N_r(4(k+1)n + 4n + 6) + 3n$$

operations are required by the orthogonalizations in total. Similarly, to compute q_{k+3} on lines 6 and 7 we need a matrix-vector product and an orthogonalization against a basis of size $k+2$, i.e.,

$$S_2 = N_r(4(k+2)n + 4n + 6) + 3n$$

operations are needed for orthogonalizations. As before, to compute a basis for the two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$, only $\frac{1}{2}(m+1)$ steps of the iteration are needed in total. After denoting $k = 2j-1$, the computational cost of generating a two-sided subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ with Algorithm 4.1 becomes

$$\sum_{j=1}^{(m+1)/2} \big(S_1 + S_2\big) = O(N_r(2nm^2 + 14nm + 12n) + 3nm + 3n)$$

operations for orthogonalizations and $m+2$ matrix-vector products in total.

In addition to solving a $(m+3)$-by-$(m+2)$ least-squares problem, the cost of which is negligible, to compute the approximate solution x_m, Algorithm 4.1 needs to apply the operator S^{-1} once. No other additional operations, such as computing the representation of the vectors Sq_k in the generated subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$, are needed.

Summarizing, we have Table 4.1 containing the most significant terms for constructing a subspace of size $(m+3)$ and obtaining an approximate solution x_m, for m odd. As mentioned already at the beginning of this section, since typically $m \ll n$, the computational cost for the solution of a $(m+3)$-by-$(m+2)$ least-squares problem is of the order $O(m^2)$ when plane rotations are used and is therefore neglible.

Complexities for different implementations of GMRES in Table 4.1 were obtained from [105]. It was assumed that the initial guess x_0 equals an all-zero vector. Flops refers to the number of operations required for orthogonalizations, that is, cost to orthonormalize $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ for TSMRES and $\mathcal{K}_{m+2}(S; q_1)$ for GMRES. Mvp refers to the number of matrix-vector products used. Mem denotes the amount of storage required.

	TSMRES	TSMRES, Prop. 4.2	Stabilized TSMRES	GMRES	Householder GMRES
Flops	$3nm^2$	$2nm^2$	$2N_r nm^2$	$2nm^2$	$4nm^2$
Mvp	$\frac{3}{2}(m+1)+1$	$m+2$	$m+3$	$m+2$	$m+2$
Mem	$(m+3)n$	$(m+3)n$	$(m+3)n$	$(m+3)n$	$(m+3)n - \frac{1}{2}m^2$

Table 4.1: Computational complexity of TSMRES versus GMRES, where N_r denotes the number of orthogonalization steps

From Table 4.1 we observe that TSMRES without the use of Proposition 4.2 is computationally slightly more expensive than GMRES. With Proposition 4.2 the computational costs are of the same order of magnitude assuming that operating with S and S^{-1} is equally expensive.

Similar conclusions hold also to the numerically stabilized TSMRES method, i.e., Algorithm 4.1. When $N_r = 1$, that is, only a single orthogonalization step is used to orthogonalize each basis vector constructed, the number of the operations required by orthogonalizations is of the same order of magnitude. The additional matrix-vector required by the numerically stabilized TSMRES method arises in the computation of the approximate solution.

We can also observe that the Householder implementation of GMRES requires about twice the number of floating point operations to construct an orthonormal basis for the subspace $\mathcal{K}_{m+2}(S; q_1)$ when compared to the implementations using Gram-Schmidt process. When Algorithm 4.1 is used with two orthogonalization steps, the cost is similar to the Householder GMRES method.

The memory usage of the different versions of the TSMRES and GMRES methods is essentially equal. The small gain for the Householder version of GMRES is explained by the decreasing length of the Householder vectors.

4.7 Conclusions

In this chapter, we have considered the implementation of TSMRES. In particular, efficient solution of the minimization problem (2.20) and the con-

struction of a representation of the matrix S in the basis generated for the two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ were presented. When solving very large linear systems with Algorithms 2.2 and 4.1, these implementation details are especially important.

We also introduced a new version of TSMRES, implemented by using iterative modified Gram-Schmidt process and reformulated minimization problem. In numerical experiments, we noticed that the numerically stabilized TSMRES was, in some cases, numerically more stable than the TSMRES method described by Huhtanen and Nevanlinna in [71].

We then briefly analyzed the numerical properties of both implementations of TSMRES. Although no backward stability was established, we gave some theoretical justification on why the numerical behaviour of the numerically stabilized TSMRES method is better than that of the standard TSMRES method.

Finally, we analyzed the computational complexity and memory consumption of TSMRES. It was shown, that when properly implemented the computational cost of both versions of TSMRES is equal to that of GMRES, assuming operating with S^{-1} is computationally as expensive as operating with S. Thus, in the cases where the convergence of TSMRES is better than that of GMRES, TSMRES can be a viable alternative.

Chapter 5

Restarting and preconditioning of TSMRES

In this chapter, we consider the use of TSMRES for solving linear systems of the form (2.1) in practice. To address the growing computational complexity and memory consumption of the full version of TSMRES, we present a restarted version of the algorithm. Another option is to use a truncated version of the recurrence similarly to DQGMRES [108], for instance, but here we do not pursue this further. For subspaces of similar size we present several cases where the restarted GMRES method stagnates, but the restarted TSMRES method converges.

For solving the standard linear system (1.1), we consider splitting the matrix A into two readily invertible parts L and R. Then, after multiplying by the inverse L of the matrix (or R), the linear system is the form (2.1) with $S = L^{-1}R$ (or $S = R^{-1}L$). The operation of splitting and multiplying by the inverse a readily invertible part can be considered as preconditioning a linear system for TSMRES. The aim is to choose L and R such that the spectrum of $S = L^{-1}R$ benefits the TSMRES method.

We begin by presenting a restarted version of TSMRES. We also shortly consider the effects of restarts with the two-sided Krylov subspaces.

5.1 Restarted TSMRES

The computational complexity analysis of Section 4.6 (see Table 4.1) shows that the amount of computational work required for the orthogonalization of the basis vectors grows quadratically as TSMRES iteration proceeds. To compute a subspace of the size $m + 2$ with the TSMRES method approximately $O(m^2 n)$ operations are required in total. Regarding memory, at each

step of the iteration two new basis vectors have to be stored. Thus, for a subspace of the size $m+2$, the storage of the basis vectors requires $O((m+2)n)$ memory. In addition to the basis vectors, a full $(m+3)$-by-$(m+2)$ block Hessenberg matrix has to be stored.

Now consider the case where n is very large and assume that the products Sv and $S^{-1}v$ for vectors $v \in \mathbb{C}^n$ can be computed in approximately $O(kn)$ operations. For $m^2 > k$ the computational cost of the orthogonalization process becomes the dominating cost. Also, for very large m, the memory requirement for the storage of the basis vectors may become a prohibitive factor.

As with the GMRES method, a simple remedy to the problem of the growing orthogonalization cost and memory consumption is to stop the TSMRES method after m steps have been taken. The method is then restarted by using the mth iterate as the new initial guess, i.e., setting $x_0 = x_m$.

In the following, we denote by N_k the maximum number of restarts and r_{tol} the stopping tolerance. A restarted version of TSMRES is described in Algorithm 5.1.

Algorithm 5.1 Restarted TSMRES

1: **for** $k = 1, 2, \ldots, N_k$ **do**
2: Lines 1-17 of Algorithm 2.2
3: If $||b - (I+S)x_m||_2/||b||_2 \leq r_{\text{tol}}$ stop, otherwise set $x_0 = x_m$
4: **end for**

We denote Algorithm 5.1, i.e., the TSMRES method restarted every m iterations, by TSMRES(m). We note that each inner cycle of TSMRES(m) contains m iterations of TSMRES for a subspace size of $2m+1$ at most. For N_k restarts, at most $N_k m$ iterations and the computation of $N_k(2m+1)$ matrix-vector products in total are required.

Denote by $r_k^{(m)}$, $1 \leq k \leq N_k m$ the kth residual vector of the restarted TSMRES method. In the absence of the numerical error, the norm of the restarted residual vector is nonincreasing by the minimum residual property of TSMRES. Thus we have $||r_k^{(m)}||_2 \leq ||r_j^{(m)}||_2$ for all $j < k$. Note that with $m = n$, we have the unrestarted full method.

Analogously to the restarted GMRES method, the problem with the restarted TSMRES is the loss of optimality. For the restarted TSMRES we have $||r_k^{(n)}||_2 \leq ||r_k^{(m)}||_2$ with $k > m$, i.e., the speed of convergence of the restarted method may be slower than that of the full method, at least when measured by the amount of iterations computed. In practice the wall-clock time taken by the computations of the restarted TSMRES method may

actually be much lower due to the reduced computational cost of orthogonalization, however.

On the other hand, as with the restarted GMRES method, it is very difficult to consider how the restarted TSMRES behaves for different values of the restart parameter m. A natural assumption would be that increasing m leads to a faster convergence. However, for the restarted GMRES method, it is well known that there are cases where $||r_k^{(m)}||_2 > ||r_k^{(l)}||_2$ with $l < m < k$, [42], i.e., increasing the restart parameter actually slows down convergence.

A more serious matter with the restarted TSMRES method is that it can stagnate, i.e., achieve no reduction of the residual norm after a restart has been made. Then $||r_l^{(m)}||_2 = ||r_k^{(m)}||_2$ for all $1 \leq l \leq k \leq N_k m$ and the restarted method fails to converge. This happens when essentially the same two-sided Krylov subspace is generated before and after a restart.

Let the GMRES-S^{-1} method be defined as in Section 2.5. We denote by GMRES(m) and GMRES-S^{-1}(m) the GMRES and GMRES-S^{-1} methods restarted after m steps, respectively. Then, for the convergence of the restarted TSMRES method, we have the following proposition.

Proposition 5.1. *Let S be a positive definite matrix. Then the restarted* TSMRES *method converges for any $m \geq 1$.*

Proof. If the matrix S is positive definite, then also then matrix S^{-1} is positive definite. The two-sided Krylov subspace generated by TSMRES(m) equals $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1) = \text{span}_{\uparrow, \uparrow \in \mathbb{P}_{\lceil \frac{m}{2} \rceil}} \{\uparrow(S)q_1, \uparrow(S^{-1})q_1\}$ where $q_1 = r_0/||r_0||_2$ and $\mathbb{P}_j$ denotes the set of polynomials of degree j at most.

With GMRES($\lceil \frac{m}{2} \rceil$) a subspace $\mathcal{K}_{\lceil \frac{m}{2} \rceil}(S; q_1) = \text{span}_{\uparrow \in \mathbb{P}_{\lceil \frac{m}{2} \rceil}} \{\uparrow(S)q_1\}$ is generated and we have $\mathcal{K}_{\lceil \frac{m}{2} \rceil}(S; q_1) \subset \mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$. On the other hand, with GMRES-S^{-1}($\lceil \frac{m}{2} \rceil$) a subspace $\mathcal{K}_{\lceil \frac{m}{2} \rceil}(S^{-1}; q_1)$ is generated, and again we have $\mathcal{K}_{\lceil \frac{m}{2} \rceil}(S^{-1}; q_1) \subset \mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$.

When S is positive definite, both GMRES($\lceil \frac{m}{2} \rceil$) and GMRES-$S^{-1}$($\lceil \frac{m}{2} \rceil$) converge, see [105, Theorem 6.30, pp. 205]. TSMRES(m) minimizes the residual norm in the two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$, which contains the subspaces $\mathcal{K}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ and $\mathcal{K}_{\lceil \frac{m}{2} \rceil}(S^{-1}; q_1)$. Therefore, the residual norm of the TSMRES(m) method before a restart must be at least as good as the minimum of the residual norms of GMRES($\lceil \frac{m}{2} \rceil$) and GMRES-$S^{-1}$($\lceil \frac{m}{2} \rceil$) before a restart and thus the TSMRES(m) method converges. □

Several methods have been proposed to prevent the stagnation and improve the convergence of the restarted GMRES method. These include augmenting the subspace with approximate eigenvectors, restarting the iteration implicitly and using deflated restarting [87, 88, 89]. In the nested Krylov

methods such as GMRESR [123] and FGMRES [102], the approximation subspace is modified to achieve a convergence more similar to that of the full method without the added computational and memory cost. Many of such techniques are applicable in the restarted TSMRES method. In this chapter we only concentrate on the restarted TSMRES method without any other modifications.

Concerning the numerical stability, the restarted TSMRES method has some benefits over the full method. Since generally $m \ll n$, the maximum dimension of the two-sided subspace is moderate and thus the degree of the polynomials $\mathfrak{p}_k$ and $\mathfrak{q}_k$ in S and S^{-1} in the relation (2.26) is also moderate. Thus the numerical stability is less of a concern. For the restarted TSMRES method, the numerical stabilization is not usually needed and all versions of TSMRES can be used. Regarding the computational complexity, this means that the number of matrix-vector products required by restarted TSMRES is equal to that required by the restarted GMRES.

In the following section, we discuss the numerical behaviour of the restarted TSMRES method in more detail.

5.2 Convergence of restarted TSMRES

In this section, we reconsider the numerical examples of Section 2.6. In the numerical experiments, we consider how the convergence of the restarted TSMRES compares to that of the restarted GMRES or GMRES-S^{-1} methods for different values of the restart parameter m.

Denote by GMRES(k) and GMRES-S^{-1}(k), the GMRES and GMRES-S^{-1} methods restarted after k steps, respectively. Then, after setting $k = 2m + 1$, the dimension of the standard Krylov subspaces $\mathcal{K}_{2m+1}(S; q_1)$ and $\mathcal{K}_{2m+1}(S^{-1}; q_1)$ used by GMRES(k) and GMRES-S^{-1}(k) equals to that of the two-sided Krylov subspace $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ used by TSMRES(m).

In the numerical experiments, the restart parameter is chosen as $m = 3, 5, 15$ or 30. The right-hand side is always $b = \hat{b}/||\hat{b}||_2$, where $\hat{b} = \mathbf{1} = (1, \ldots, 1)^T$ is the vector of all ones. As a stopping criterion

$$\frac{||b - (I + S)x||_2}{||b||_2} \leq 1E - 6 \tag{5.1}$$

is used, i.e., we compute the residual from scratch each iteration round and terminate the iteration once the relative residual is reduced by six orders of magnitude. The maximum number of restarts N_k is chosen such that approximately 2000 matrix-vector products are computed at most.

For Examples from 2.6 to 2.21 of Section 2.6, we describe the number of the matrix-vector products required by the restarted GMRES, GMRES-S^{-1} and TSMRES methods to reach the stopping criterion (5.1) for different values of the restart parameter m. A dagger-symbol (†) indicates that the stopping criterion is not reached within a maximum number of matrix-vector products. Stagnation of the iteration is denoted by a double dagger-symbol (‡). A bold value indicates the fastest convergence among the three methods for each example–restart parameter pair. Table 5.1 describes the results.

	GMRES(k)				GMRES-S^{-1}(k)				TSMRES(m)			
	7	11	31	61	7	11	31	61	3	5	15	30
Ex 2.6	59	51	48	46	**8**	**8**	**8**	**8**	17	16	15	15
Ex 2.7	**11**	**11**	**11**	**11**	1071	516	116	80	19	20	21	21
Ex 2.8	632	389	200	172	**11**	**10**	**10**	**10**	24	22	21	21
Ex 2.9	**51**	**45**	**43**	**42**	†	†	†	†	115	99	83	84
Ex 2.10	†	†	†	†	†	†	**459**	**104**	†	†	†	1025
Ex 2.11	146	115	91	90	**34**	**28**	**21**	**21**	66	58	36	35
Ex 2.12	**153**	**145**	138	136	419	230	**96**	**68**	257	231	149	110
Ex 2.13	†	†	†	†	**20**	**20**	**20**	**20**	47	44	42	41
Ex 2.14	**20**	**20**	**20**	**20**	†	†	†	†	35	36	38	39
Ex 2.15	†	†	†	†	†	†	†	†	†	†	†	†
Ex 2.16	20	20	20	20	**11**	**11**	**11**	**11**	17	16	15	15
Ex 2.17	**11**	**11**	**11**	**11**	21	21	21	21	14	14	15	15
Ex 2.18	†	†	†	†	**149**	**110**	**79**	**60**	493	379	191	163
Ex 2.19	†	1383	761	669	**16**	**16**	**16**	**16**	38	36	31	31
Ex 2.20	‡	‡	†	†	‡	‡	†	†	‡	‡	†	†
Ex 2.21	1443	586	**270**	**257**	1594	598	288	267	**416**	**322**	279	**257**

Table 5.1: Number of matrix-vector products required to reach the convergence criterion (5.1) for GMRES(k), GMRES-S^{-1}(k) and TSMRES(m)

The results are similar to those presented in Section 2.6. We observe that the differences in the convergence speed are emphasized when the methods are being restarted, especially for small m. For the cases considered, the restarted TSMRES is the most robust method, failing to converge only when both the GMRES and GMRES-S^{-1} methods fail to converge.

In most cases, the convergence speed of the restarted TSMRES method is between the convergence speed of the restarted GMRES and GMRES-S^{-1} methods. If either GMRES or GMRES-S^{-1} converges rapidly, in most cases TSMRES also converges rapidly, requiring approximately two times the number of matrix-vector products when compared to the fastest method. We attribute such behaviour to the construction of the two-sided subspaces $\mathcal{K}^{\pm}_{\lceil \frac{m}{2} \rceil}(S; q_1)$ and the minimum residual property of TSMRES. In such cases, approximately half of the generated set of basis vectors cannot be efficiently used in reducing the residual. The phenomenon is observable in Examples 2.6 and 2.14, for instance.

None of the restarted methods converge for Example 2.15 or 2.20. By the construction of $\sigma(S)$ in these examples, such behaviour is to be expected, since the approximation with polynomials π_j or q_j in S or S^{-1} is not efficient until the degree of the polynomials has reached n or $n/2$, respectively. Thus, for these examples, restarting the iteration yields very poor convergence.

For the full methods, in Example 2.21 of Section 2.6, TSMRES converges faster than GMRES and GMRES-S^{-1} by a small margin. For the same example with the restarted algorithms, TSMRES is usually the most efficient method by a more significant margin, especially when m is small. Approximately a thousand matrix-vector products less are required by TSMRES for $m = 3$ when compared to GMRES and GMRES-S^{-1}. For $m = 5$, the reduction is reduced to about a two hundred matrix-vector products. For larger m, all the methods perform almost equally, reaching the convergence criterion (5.1) within a twenty matrix-vector products of each other.

From the numerical results of this section, we conclude that when the spectrum of S resembles that of Example 2.21, the restarted TSMRES is a viable alternative to the restarted GMRES methods. In the next section, we consider preconditioning a general linear system (1.1) to the form (2.1).

5.3 Preconditioning of general linear systems for the TSMRES method

In this section, we consider solving the linear system (1.1), i.e,

$$Ax = b$$

with $A \in \mathbb{C}^{n\times n}$ and $b \in \mathbb{C}^n$. We now assume that there exists a splitting of the matrix A of the form

$$A = L + R, \tag{5.2}$$

with $L, R \in \mathbb{C}^{n\times n}$ both readily invertible. Then, after preconditioning from the left with L^{-1} (or R^{-1}) we have the linear system (2.1), i.e.,

$$(I + S)x = L^{-1}b \tag{5.3}$$

with $S = L^{-1}R$ (or $S = R^{-1}L$) readily invertible. With a similar splitting, preconditioning can also be applied from the right to have

$$(I + S)y = b, \quad x = L^{-1}y, \tag{5.4}$$

with $S = RL^{-1}$ again readily invertible. In both cases, the preconditioning operation becomes cleverly splitting A into L and R with the aim of having such a spectrum for S that benefits the full or the restarted TSMRES method.

Our preconditioning approach is different from the standard approach to preconditioning. With the methods based on standard Krylov subspaces $\mathcal{K}_j(A; q_1)$, such as GMRES, the aim is usually to heuristically find a preconditioner M such that the distance of MA to the identity operator is smaller than that of the matrix A alone. Then, the preconditioned Krylov subspaces $\mathcal{K}_j(MA; q_1)$ can usually be expected to yield a faster convergence than the unpreconditioned ones.

We note that an attempt to apply the conventional left preconditioning strategy to (1.1) leads to a new decomposition of A. After applying the inverse of ML we then have

$$(ML + MR)x = Mb \Leftrightarrow (I + S)x = L^{-1}b, \tag{5.5}$$

which is equivalent to the linear system (2.1) with a modified right-hand side. An attempt to use the right preconditioning yields

$$(LM + RM)u = b, \ u = M^{-1}x \tag{5.6}$$

Multiplying by the inverse of LM from the left, we have

$$(I + M^{-1}SM)u = M^{-1}L^{-1}b, \ \Leftrightarrow M^{-1}(I + S)Mu = M^{-1}L^{-1}b, \tag{5.7}$$

i.e., a similarity transformation by the matrix M. Since the preconditioned matrix $M^{-1}(I + S)M$ has the same eigenvalues as the original matrix $I + S$, from the view of the approximation problem (2.30), the right preconditioning does not transform the system into a one that is easier to solve with TSMRES.

Now briefly consider splitting a general nonsingular matrix A with nonzero diagonal entries as $A = L + R$. The standard Gauss-Seidel splitting is of the form

$$A = D - E - F, \tag{5.8}$$

where D is the diagonal and $-E$ and $-F$ are the strictly lower and upper diagonal parts of A. In the Gauss-Seidel iteration operations are then performed with the matrices $L = D - E$ (or $L = D - F$) and $R = -F$ (or $R = -E$). With the Gauss-Seidel splitting, the matrix R does not need to be invertible.

Splitting a matrix is even more complex when both parts of the splitting must be readily invertible. A special care must be taken in order to obtain L and R which are not only nonsingular, but also not too ill-conditioned to be used in practical computations.

By our assumptions the diagonal of the matrix A is always nonzero, which may seem very restrictive at first. However, a way to deal with matrices

having zeroes on their diagonal is to nonsymmetrically permute A so that its diagonal entries become nonzero. We can also try to seek a permutation maximizing the sum (or product) of the diagonal entries. Algorithms for constructing such permutations, called the maximum weighted matchings of a bipartite graph of A in graph theoretic terms, have been studied by Duff and Koster in [37]. Benzi et al. studied their effects in the context of preconditioning iterative methods in [8]. For a nonsingular matrix A, such a matching always exists [93]. Thus it is, at least in theory, always possible to obtain an invertible Gauss-Seidel-type splitting after the matrix has been permuted to have nonzero diagonal entries.

In the following subsections, we consider solving the linear system (2.1) obtained after splitting the original matrix A into two readily invertible parts and then preconditioning the linear system (1.1) by one of the parts from the right. Therefore we have the linear system (5.4) with $S = RL^{-1}$. In exact arithmetic, the residual norms computed by GMRES, GMRES-S^{-1} and TSMRES for the preconditioned linear system (5.4) are equivalent to the residual norms computed for the original linear system. In spite of this, we use the stopping criterion

$$\frac{||b - Ax||_2}{||b||_2} \leq 1E - 6, \tag{5.9}$$

i.e., the iteration is terminated once the relative residual norm of the original linear system (1.1), computed from scratch each iteration round, has been reduced by six orders of magnitude. We note that in the numerical experiments, the computed residual of the preconditioned linear system was always of the same order of magnitude as the residual used in the stopping criterion (5.9).

In the numerical experiments, we compare the convergence speed of TSMRES(m) with that of GMRES(k) and GMRES-S^{-1}(k) where $k = 2m+1$ and $m = 5, 10$ and 15. The maximum number of restarts N_k is chosen such that approximately 2500 matrix-vector products are computed at most. As a right-hand side we use $b = \hat{b}/||\hat{b}||_2$, where $\hat{b} = \mathbf{1} = (1, 1, \ldots, 1)^T$ denotes a vector of all ones as previously.

To construct a readily invertible matrix S we split the matrix A as described in what follows. We consider two basic types of splittings: purely algebraic and ones based on the underlying equations. As purely algebraic splittings we consider the splittings of the Gauss-Seidel and the k-Hessenberg –type. As an example of a splitting based on splitting the underlying discretized equations, we study splittings of the ADI –type.

Denote by nz(X) the number of nonzero elements in a matrix X. In the numerical experiments with purely algebraic splittings, we let the matrix

A be a matrix selected from the MatrixMarket collection [92]. We use the following test problems.

- **JPWH991** is generated by circuit physics modelling. It is of the size $n = 991$ with $\mathrm{nz}(A) = 6027$. The condition number estimate is moderate $\kappa^{\mathrm{est}}(A) = 7.3E + 2$.
- **Sherman1** is from a black oil simulation of shale barriers. The matrix size is $n = 1000$ with $\mathrm{nz}(A) = 3750$ and the condition number estimate $\kappa^{\mathrm{est}}(A) = 2.3E + 4$.
- **Sherman4** is from an IMPES simulation with flow barriers. The matrix size is $n = 1104$ with $\mathrm{nz}(A) = 3786$ and the condition number estimate $\kappa^{\mathrm{est}}(A) = 7.2E + 3$.
- **Sherman5** is from a fully implicit black oil simulator. The matrix size is $n = 3312$ with $\mathrm{nz}(A) = 20793$ and the condition number estimate $\kappa^{\mathrm{est}}(A) = 3.9E + 5$.

With ADI–type of splittings we consider finite difference discretizations of elliptic equations of the type

$$-(au_x)_x - (bu_y)_y + cu_x + (cu)_x + du_y + (du_y)_y + eu = f, \tag{5.10}$$

on a rectangular region $\Omega \subset \mathbb{R}^2$. We use Dirichlet boundary conditions $u = 0$ on the boundary $\partial\Omega$. Denote by $\overline{\Omega}$ the closure of Ω, i.e., $\overline{\Omega} = \Omega \cup \partial\Omega$. Then, for the functions $a(x, y), b(x, y), c(x, y)$ and $d(x, y)$ we assume $C^1(\Omega) \cap C^0(\overline{\Omega})$ to hold, i.e., the functions are continuously differentiable inside the domain and continuous on the closure of Ω. Function $e(x, y)$ is assumed to be continuous on $\overline{\Omega}$.

For comparison, we now consider solving the linear system (1.1), i.e.,

$$Ax = b$$

for all the test problems from the MatrixMarket collection with GMRES(k). We also consider the case with a right preconditioner (1.22)

$$APy = b, \quad x = Py,$$

where we use $ILU(\tau)$ preconditioner, i.e., the incomplete LU-decomposition with threshold dropping, computed with MATLAB's `luinc`-function. With $ILU(\tau)$, we choose the threshold of the numerical dropping for the test problems **JPWH991**, **Sherman1**, **Sherman4** and **Sherman5** as $\tau = 5E -$

$1, \tau = 7E-2, \tau = 1.5E-1$ and $\tau = 7E-3$, respectively. With these choices of τ, the number of the entries in the computed incomplete factors of the preconditioner and in the original matrix is approximately the same. We use the stopping criterion (5.9), i.e., the iteration is stopped when the relative residual has been reduced by six orders of magnitude.

Table 5.2 describes the number of matrix-vector products required by the restarted GMRES method with and without $ILU(\tau)$ preconditioner to reach the stopping criterion (5.9). Operations required to apply the preconditioner were not taken into account. Nonconvergence and stagnation of the method is denoted by dagger (†) and double dagger (‡) –symbols, respectively.

	GMRES(k)			GMRES(k)-$ILU(\tau)$		
	11	21	31	11	21	31
JPWH991	73	52	43	57	45	39
Sherman1	†	†	†	100	74	69
Sherman4	697	495	351	162	114	62
Sherman5	†	†	†	74	53	33

Table 5.2: Number of matrix-vector products required to have (5.9) for GMRES(k) with and without $ILU(\tau)$ preconditioning for the linear system (1.1)

From Table 5.2 we observe that without preconditioning, only the problems **JPWH991** and **Sherman4** are relatively easy to solve with the restarted GMRES method. On the other hand, by using the $ILU(\tau)$ preconditioner, practically all of the problems become quite easy to solve.

5.3.1 Gauss-Seidel –type splittings

In this section, we consider algebraic splitting of the matrix A into block lower and upper triangular matrices. Let the matrix A be partitioned into s^2 blocks such that there are s blocks both row- and columnwise. We have

$$A = \begin{bmatrix} A_{1,1} & A_{2,1} & \cdots & A_{1,s} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,s} \\ \vdots & \vdots & \ddots & \vdots \\ A_{s,1} & A_{s,2} & \cdots & A_{s,s} \end{bmatrix},$$

which yields a block splitting of A as

$$D = \begin{bmatrix} A_{1,1} & & & \\ & A_{2,2} & & \\ & & \ddots & \\ & & & A_{s,s} \end{bmatrix}, \tag{5.11}$$

$$E = -\begin{bmatrix} \mathbf{0} & & & \\ A_{2,1} & \mathbf{0} & & \\ \vdots & \vdots & \ddots & \\ A_{s,1} & \cdots & A_{s,s-1} & \mathbf{0} \end{bmatrix}, \quad F = -\begin{bmatrix} \mathbf{0} & A_{1,2} & \cdots & A_{1,s} \\ & \mathbf{0} & \vdots & \vdots \\ & & \ddots & A_{s-1,s} \\ & & & \mathbf{0} \end{bmatrix},$$

where D is the block diagonal and $-E$ and $-F$ are the strictly lower and upper block diagonal parts of A and $\mathbf{0}$ an all-zero block matrix of an appropriate size. We assume that in the splitting all the diagonal blocks of $A_{j,j}$ are invertible, i.e., the matrix D is invertible. With D, $-E$ and $-F$ defined as in (5.11), we obtain a readily invertible block triangular L and R by setting $L = \frac{1}{2}D - E$ and $R = \frac{1}{2}D - F$.

The block triangular splittings present a connection between direct and iterative methods. As the TSMRES iteration proceeds, operations on S and S^{-1} are applied multiple times. In order to efficiently solve block triangular systems with a block forward or back substitution, we can compute and store the LU-decompositions of the diagonal blocks of A as $D_{j,j} = L_{j,j}U_{j,j}$. Then the process of solving linear systems with L and R becomes inexpensive in terms of floating point operations since the diagonal elements $\frac{1}{2}D_{j,j}$ are not repeatedly factorized.

Storing the LU-decompositions of the diagonal blocks has an additional memory cost determined by the structure of the diagonal blocks. If the blocks are sparse, fill-in reducing permutations such as the reverse Cuthill-McKee [52] or the column approximate minimum degree [30] can be used blockwise to reduce the amount of storage required.

Even if the blocks are full, assuming $l \ll n$ and that s^2 blocks of constant size l-by-l are used, storing the LU-decompositions of the diagonal blocks takes approximately $O(sl^2)$ memory. The amount is equivalent to what is required by the standard block Jacobi-preconditioner.

We now consider solving the test problems with the block lower and upper triangular splittings. If possible, we select the blocksize $l \in \mathbb{N}$ as a fixed constant such that $n = ls$ holds with $s \in \mathbb{N}$, i.e., l is selected to divide n. For the matrix **JPWH991**, the last block was always selected to be of the size $l + 1$.

As previously, we describe the number of the matrix-vector products required by the restarted GMRES, GMRES-S^{-1} and TSMRES methods to reach the stopping criterion (5.9). Nonconvergence and stagnation of the methods are denoted by dagger (†) and double dagger (‡) –symbols, respectively. A bold value indicates the fastest convergence among the three methods for each example–restart parameter pair. Table 5.3 describes the results for each test problem for different blocksizes l with the restarted GMRES(k), GMRES-S^{-1}(k) and TSMRES(k) methods for $m = 5, 10$ and 15 and $k = 2m + 1$.

	GMRES(k)			GMRES-S^{-1}(k)			TSMRES(m)		
	11	21	31	11	21	31	5	10	15
JPWH991									
$l = 1$	**82**	57	49	86	**55**	49	84	57	**44**
$l = 5$	**82**	57	49	85	**56**	49	84	57	**44**
$l = 11$	**83**	56	48	82	**54**	48	84	55	**44**
Sherman1									
$l = 1$	‡	‡	‡	‡	‡	‡	‡	‡	‡
$l = 10$	216	134	**114**	316	**123**	118	**174**	131	120
$l = 20$	154	**98**	100	212	**98**	95	**135**	103	**93**
Sherman4									
$l = 1$	‡	714	332	‡	‡	515	‡	**418**	**323**
$l = 6$	‡	451	399	‡	‡	362	‡	**402**	**302**
$l = 16$	‡	376	337	‡	‡	331	‡	**360**	**290**
Sherman5									
$l = 1$	‡	‡	‡	‡	‡	‡	‡	‡	†
$l = 69$	‡	†	327	‡	†	**264**	‡	‡	313
$l = 144$	‡	495	184	‡	293	179	‡	**269**	**170**

Table 5.3: Number of matrix-vector products required to have (5.9) for GMRES(k), GMRES-S^{-1}(k) and TSMRES(m) for the linear system (2.1) obtained after a block triangular splitting.

By the numerical results of Table 5.3, we observe that for the test problem **JPWH991** all the methods perform nearly equally. For the test problem **Sherman1**, when the methods do not stagnate, TSMRES is the fastest method when the maximum size of the subspace is 11. Also, for the test problem **Sherman4**, the TSMRES method is the fastest method in all cases when the methods do not stagnate. The test problem **Sherman5** is difficult to solve with the block triangular splitting approach. Large sized blocks and

large sized subspaces are required to prevent stagnation.

In the cases where TSMRES is the fastest method, its convergence speed is not competitive with ILU preconditioning. By the results of Table 5.2, the iteration counts for TSMRES are larger than those of the restarted GMRES method with an $ILU(\tau)$ preconditioner applied to the original linear system (1.1).

We note that all of the **ShermanX** test problems are structured, i.e., they arise from finite difference discretizations. In the numerical experiments, we selected the block sizes for these problems according to the parameters used in the discretization. For a black-box method such information is not always directly available, but can be extracted automatically, see [39].

With the restarted TSMRES method, increasing the block size l is beneficiary to the convergence speed of the iteration for all of the test problems. When the problem is structured, also the restarted GMRES and GMRES-S^{-1} methods benefit from the increased block size. When the test problem is unstructured, the benefits of the block splitting approach are not so clear. For all the methods the convergence for the test problem **JPWH991** is largely similar even if the block size is increased.

By the numerical results, we conclude that preconditioning linear systems for TSMRES by using block splittings seems to be applicable at least in the case where the linear system has some predefined block structure. On the other hand, using an $ILU(\tau)$ preconditioner on the original linear system can yield much faster convergence. In the next section, we consider splittings better suited to cases where the structure of the matrix is not known beforehand.

5.3.2 k-Hessenberg splittings

In this section we consider algebraic splitting of the original matrix A into lower and upper k-Hessenberg matrices. By an upper k-Hessenberg matrix we mean an upper triangular matrix with k extra nonzero diagonals below the main diagonal. Lower k-Hessenberg matrices are defined similarly as lower triangular matrices having k extra nonzero diagonal above the main diagonal.

In the following we partition the matrix A into lower and upper k-Hessenberg parts. Denote by $a_{i,j}$ the entries of the matrix A. We then have

$$A = \begin{bmatrix} a_{1,1} & a_{2,1} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix},$$

which yields a k-band splitting of A as

$$B = \begin{bmatrix} a_{1,1} & \cdots & a_{1,k+1} & & & & \\ \vdots & \ddots & & \ddots & & & \\ a_{k+1,1} & & \ddots & & \ddots & & \\ & \ddots & & \ddots & & a_{n-k-1,n} & \\ & & \ddots & & \ddots & \vdots & \\ & & & a_{n,n-k-1} & \cdots & a_{n,n} & \end{bmatrix}, \tag{5.12}$$

$$E = -\begin{bmatrix} & & & \\ a_{k+2,1} & & & \\ \vdots & \ddots & & \\ a_{n,1} & \cdots & a_{n,n-k-2} & \end{bmatrix}, F = -\begin{bmatrix} a_{1,k+2} & \cdots & a_{1,n} \\ & \ddots & \vdots \\ & & a_{n-k-2,n} \\ & & \end{bmatrix},$$

where B is the k-band and $-E$ and $-F$ are the strictly lower and upper k-diagonal parts of the matrix A. With these, we have the k-Hessenberg splitting of A as $L = \frac{1}{2}B - E$ and $R = \frac{1}{2}B - F$.

Linear systems involving sparse k-Hessenberg matrices are numerically inexpensive to solve. Consider a linear system

$$Hx = b, \tag{5.13}$$

where $H \in \mathbb{C}^{n\times n}$ denotes an upper (or lower) k-Hessenberg matrix and $b \in \mathbb{C}^n$. Denote by $\tau = \mathrm{nz}(H)/n^2$ the density of the matrix H. Then, solving the linear system (5.13) requires $O(\tau(k+1)n^2)$ floating point operations and $O(kn)$ memory. In Chapter 6, we present an algorithm having such a computational complexity in detail.

Compared to the block triangular splittings, the k-Hessenberg splittings have the advantage of requiring less knowledge about the structure of the original matrix. On the other hand, nonsingularity of the k-Hessenberg parts can be much harder to determine.

Table 5.4 describes the results of the numerical experiments with k-Hessenberg splittings for several different values of the parameter k. As with the block triangular splittings, we describe the number of matrix-vector products required by the restarted GMRES($\hat{k}$), GMRES-S^{-1}($\hat{k}$) and TSMRES(m) methods for $m = 5, 10$ and 15 and $\hat{k} = 2m+1$ to reach the stopping criterion (5.9). As before, nonconvergence and stagnation are denoted by dagger (†) and double dagger (‡) –symbols. A bold value indicates the fastest convergence among the three methods for each example–restart parameter pair.

	GMRES($\hat{k}$)			GMRES-$S^{-1}(\hat{k})$			TSMRES(m)		
	11	21	31	11	21	31	5	10	15
JPWH991									
$k = 0$	**82**	57	49	86	**55**	49	84	57	**44**
$k = 1$	**82**	57	49	86	**56**	49	84	57	**46**
$k = 5$	83	56	48	**82**	**55**	48	84	**55**	**44**
Sherman1									
$k = 0$	‡	‡	‡	‡	‡	‡	‡	‡	‡
$k = 1$	216	134	**114**	316	**123**	118	**174**	131	120
$k = 11$	101	79	66	103	80	**65**	**95**	**78**	**65**
Sherman4									
$k = 0$	‡	714	332	‡	‡	515	‡	**418**	**323**
$k = 1$	‡	376	337	‡	‡	331	‡	**360**	**290**
$k = 16$	‡	226	95	‡	226	105	**247**	**156**	**91**
Sherman5									
$k = 0$	‡	‡	‡	‡	‡	‡	‡	‡	†
$k = 1$	‡	‡	‡	‡	‡	‡	‡	‡	‡
$k = 5$	‡	‡	1738	‡	‡	**394**	‡	‡	463

Table 5.4: Number of matrix-vector products required to have (5.9) for GMRES($\hat{k}$), GMRES-$S^{-1}(\hat{k})$ and TSMRES(m) for the linear system (2.1) obtained after a k-Hessenberg splitting.

We select the band sizes k by finding the nonzero diagonals of the test matrices. For the test problems **Sherman1** and **Sherman4**, the values selected correspond to the smallest ones which yield nonzero diagonals. With **JPWH991** and **Sherman5** we use the those diagonals for which the band size k is suitably small. Splitting the matrix A with the band size $k = 0$ corresponds to splitting the matrix into lower and upper triangular parts. Thus the results for each test problem with $k = 0$ are identical to those presented in Table 5.3 with block size $l = 1$. The results were included to make comparison easier.

By the numerical experiments increasing the number of diagonals in the matrix B also increases the speed of convergence when the test problem is structured. With the test problem **JPWH991**, increasing the band size does not yield any notable increase in the speed of convergence for any of the methods tested. For the **ShermanX** test problems, increasing the band size k generally improves convergence.

In general, the convergence behaviour is very similar to increasing the

block size l with the Gauss-Seidel –type of splittings. As with the block triangular splittings, in most cases the iteration counts for TSMRES with k-Hessenberg splittings are larger than those of the restarted GMRES method with an $ILU(\tau)$ preconditioner applied to the original linear system (1.1). The exception is the **Sherman1** test problem with $m = 5$ and $k = 11$, where the speed of convergence obtained with TSMRES is faster.

The conclusions are largely similar to those of the previous section. For the structured problems, splittings of the k-Hessenberg type can be used to precondition linear systems for TSMRES. For the test problems, the restarted TSMRES method is a bit more robust, since it does not stagnate for the Sherman4 test problem when $k = 16$ and $m = 5$. For all the methods tested, increasing the band size k is beneficiary to the convergence speed of the iteration for the structured test problems.

5.3.3 ADI –type splittings

In this section, we consider splittings of the ADI–type for the elliptic equations (5.10). Alternating Direction Implicit (ADI)-iterations were originally introduced by Peaceman and Rachford in [96] (see also [125]). ADI -iterations are based on splitting the two-dimensional discretization by its x- and y-directions to have a linear system of the form

$$(H + V)x = b, \tag{5.14}$$

where H and V are large and sparse matrices having a special structure.

With the splitting (5.14), the ADI method can be formulated as

$$\begin{aligned} (\varphi I + H)x^{(j+1/2)} &= (\varphi I - V)x^{(j)} + b, \\ (\psi I - V)x^{(j+1)} &= (\psi I + H)x^{(j+1/2)} - b, \end{aligned} \tag{5.15}$$

where $\varphi, \psi \in \mathbb{R}$ are parameters which can vary each step. Let E and F be disjoint compact sets such that $\sigma(V) \subseteq E$ and $-\sigma(H) \subseteq F$, where $\sigma(V)$ and $\sigma(H)$ denote the spectrum of V and H, respectively. Then, for the commutative case $HV = VH$, optimal choice for the sets of parameters $\{\varphi_1, \varphi_2, \ldots, \varphi_l\}$ and $\{\psi_1, \psi_2, \ldots, \psi_l\}$ leads to a rational minimization problem

$$\min_{r \in R_{ll}} \frac{\max_{z \in E} |r(z)|}{\min_{z \in F} |r(z)|}, \tag{5.16}$$

where R_{ll} denotes the set of rational functions with a denominator and numerator degree not exceeding l.

In the commutative case, an algorithm for computing an optimal set of parameters via the approximation problem (5.16) was given in [126]. This

was later found to be linked to the one of the four classical approximation problems solved by Zolotarev with elliptic functions, see [114].

In the nonsymmetric case the problem of choosing optimal parameters is considerably more difficult. An asymptotically optimal approach is to use Leja points in solving the nonsymmetric equivalent of (5.16). Then the asymptotic behaviour is reached with relatively few parameters [111]. The ADI method can also be used as an effective preconditioner for an iterative method, such as GMRES, see [112].

In the following, we consider using the splitting (5.14) to transform the linear system (1.1) into the form (2.1), i.e., we do not consider the use of the parameters φ or ψ. We consider the following two test problems.

1. The Poisson's equation

$$\begin{cases} -\triangle u &= f, \quad \text{in } \Omega, \\ u &= 0, \quad \text{on } \partial\Omega, \end{cases}$$

   which is obtained from (5.10) by setting $a(x,y) = b(x,y) = 1$ and $c(x,y) = d(x,y) = e(x,y) = 0$.

2. The nonsymmetric problem

$$\begin{cases} -\triangle u + \tau(xu_x + yu_y) + \eta u &= f, \quad \text{in } \Omega, \\ u &= 0, \quad \text{on } \partial\Omega, \end{cases}$$

   with $\tau = 100$ and $\eta = -50$. The problem is obtained from the equation (5.10) by setting $a(x,y) = b(x,y) = 1$, $c(x,y) = \frac{\tau}{2}x$, $d(x,y) = \frac{\tau}{2}y$ and $e(x,y) = \eta - \tau$.

Let $\Omega = (0,1) \times (0,1)$ be the unit square and $f(x,y) = 1$. Using centered difference approximations $u_x = \partial_x u \approx \frac{u_{i+1,j} - u_{i-1,j}}{2h}$ and $u_y = \partial_y u \approx \frac{u_{i,j+1} - u_{i,j-1}}{2h}$ to the derivatives we have the discretized linear system on a n-by-n point grid with the splitting (5.14) where H and V are n^2-by-n^2 matrices. For matrices H and V we then have

$$\begin{aligned} [H]_{i,j} &= \left(a_{i+1/2,j} + a_{i-1/2,j}\right) u_{i,j} - a_{i+1/2,j} u_{i+1,j} - a_{i-1/2,j} u_{i-1,j} \\ &\quad + \left[\left(c_{i,j} + c_{i+1,j}\right) u_{i+1,j} - \left(c_{i,j} + c_{i-1,j}\right) u_{i-1,j}\right] h/2 + e_{i,j} u_{i,j} h^2/2 \end{aligned} \tag{5.17}$$

and

$$\begin{aligned} [V]_{i,j} &= \left(b_{i,j+1/2} + b_{i,j-1/2}\right) u_{i,j} - b_{i,j+1/2} u_{i,j+1} - b_{i,j-1/2} u_{i,j-1} \\ &\quad + \left[\left(d_{i,j} + d_{i,j+1}\right) u_{i,j+1} - \left(d_{i,j} + d_{i,j-1}\right) u_{i,j-1}\right] h/2 + e_{i,j} u_{i,j} h^2/2, \end{aligned} \tag{5.18}$$

	GMRES(k)			GMRES(k)-$ILU(0)$		
	11	21	31	11	21	31
Problem 1						
16×16	63	26	25	14	14	14
32×32	303	148	90	28	24	23
64×64	1088	621	458	119	52	42
128×128	4189	2258	1581	408	243	133
Problem 2						
16×16	‡	‡	214	20	16	16
32×32	‡	‡	‡	74	37	26
64×64	‡	‡	‡	‡	232	115
128×128	†	†	†	‡	‡	‡

Table 5.5: Number of matrix-vector products required to have (5.9) for GMRES(k) with and without $ILU(0)$ preconditioning for the linear system (5.14)

with $h = 1/(n+1)$ and where $a_{i,j}, b_{i,j}, c_{i,j}, d_{i,j}$ and $e_{i,j}$ denote the values of the functions a, b, c, d and e in the grid point (i, j).

For comparison, we now consider solving the linear system (5.14) by using GMRES(k) with and without a preconditioning. We use $ILU(0)$, i.e., the incomplete LU-decomposition with no fill-in as a preconditioner, computed with MATLAB's luinc-function. As a stopping criterion (5.9) is used, i.e., the iteration is stopped when the relative residual has been reduced by six orders of magnitude.

Table 5.5 describes the number of matrix-vector products required by the restarted GMRES method without and with $ILU(0)$ preconditioning to reach the stopping criterion (5.9). As before, operations required to apply the preconditioner were not taken into account. Nonconvergence and stagnation of the method is denoted by dagger (†) and double dagger (‡) –symbols, respectively.

Now consider preconditioning the linear system (5.14) with the splitting approach. When defined as in (5.17), the matrix H is block diagonal with tridiagonal blocks of size n. Similarly, with (5.18) the matrix V is block tria-diagonal with diagonal blocks. After reordering the matrix V becomes block tridiagonal with a structure similar to that of the matrix H. Since solving linear systems with tridiagonal matrices requires approximately $O(8n)$ operations [55], both H and V are readily invertible. Setting $L = H$ and $R = V$, we then have the linear system (5.4).

Table 5.6 describes the results of the numerical experiments. For each of the test problems, we describe the number of the matrix-vector products required by the restarted GMRES(k), GMRES-$S^{-1}(k)$ and TSMRES(m) methods for $m = 5, 10$ and 15 and $k = 2m+1$ to reach the stopping criterion (5.9) for different sizes of an $n \times n$ finite difference grid. As before, we denote the nonconvergence by a dagger (†) and the stagnation by a double dagger (‡) –symbols. A bold value again indicates the fastest convergence among the three methods for each grid size–restart parameter pair

	GMRES(k)			GMRES-$S^{-1}(k)$			TSMRES(m)		
	11	21	31	11	21	31	5	10	15
Problem 1									
16×16	**47**	**30**	**27**	48	33	**27**	91	38	31
32×32	**248**	124	90	**248**	**123**	**89**	300	185	141
64×64	969	**515**	**386**	**968**	**515**	**386**	1070	607	449
128×128	†	**2002**	**1376**	†	**2002**	**1376**	†	2222	1555
Problem 2									
16×16	**267**	**189**	143	‡	185	**123**	311	271	224
32×32	**316**	**273**	**231**	‡	‡	403	372	391	372
64×64	666	**567**	**556**	‡	‡	‡	**550**	588	678
128×128	‡	‡	**1099**	‡	†	†	**1184**	**1050**	1147

Table 5.6: Number of matrix-vector products required to have (5.9) for GMRES(k), GMRES-$S^{-1}(k)$ and TSMRES(m) for the linear system (2.1) obtained after an ADI-splitting

The first test problem is symmetric and positive definite and the condition $HV = VH$ holds. In addition, since the matrices H and V correspond to the discretizations of the x- and y-directions on a square finite difference grid, there exists permutation matrix P such that $H = PVP^{-1}$. Therefore H and V have the same eigenvalues and we can expect GMRES and GMRES-S^{-1} to perform almost equally, as seen from the results of Table 5.6.

For the first test problem, the convergence speed of the TSMRES method is comparable to that of the GMRES and GMRES-S^{-1} methods. Since the preconditioned matrix $I + S$ is positive definite, both version of the restarted GMRES method and the TSMRES method can be expected to converge. The convergence speed is slow, however.

A comparison with Table 5.5 reveals, that for the first problem the ILU preconditioner yields the fastest convergence. Note that due to the additional storage required by the preconditioner, when compared with the splitting approach, the memory usage is approximately doubled.

The results for the second problem, which is nonsymmetric and indefinite, are more interesting. When $k = 11$ or $k = 21$, for large grids both GMRES

and GMRES-S^{-1} stagnate. As the grid size increases, for the second test problem the restarted TSMRES method does not stagnate for any tested values of the restart parameter m.

With the TSMRES method, the number of iterations required for convergence increases as the grid size increases. When the grid size is small, 32×32 for instance, increasing the subspace size actually slows down convergence. Thus we have an example of the restarted TSMRES method behaving similarly to the restarted GMRES method, see [42].

For the second test problem, the restarted GMRES method used on the original linear system (5.14) does not converge without a large subspace. As the gridsize increases, a similar conclusion seems to hold also with the $ILU(0)$ preconditioned GMRES(k). For the largest grid tested, i.e., 128×128, the TSMRES method applied to the preconditioned linear system (2.1) is the only method that converges for $m = 11$ and $m = 21$.

5.4 Conclusions

In this chapter we have studied the convergence properties of the restarted TSMRES method. We also considered preconditioning general linear systems for TSMRES. To have a linear system with a structure suitable for TSMRES, we considered splitting the original matrix into two readily invertible parts and then preconditioning by one of the parts. This can be interpreted as preconditioning a linear system for TSMRES.

In the numerical experiments, the restarted TSMRES method was more robust method than the restarted GMRES method. In the numerical examples we presented cases where the restarted GMRES methods stagnated but the restarted TSMRES method converged. In some cases the convergence speed of restarted TSMRES was faster than that of the restarted GMRES methods, especially for small values of the restart parameter m.

The purpose of our numerical experiments was to show that it is possible to apply the splitting approach to linear systems arising in practice. However, initial comparisons with ILU preconditioners were not always favourable to the splitting approach when block triangular and k-Hessenberg splittings were used. More extensive numerical testing is needed to conclude if the approach presented in this section is competitive with the existing techniques in actual large-scale computations.

Preconditioning general linear systems for TSMRES remains a challenge, especially for highly nonsymmetric and indefinite problems. Obtaining splittings with invertible parts may be difficult due to issues such as having zeroes on the diagonal. In preliminary numerical experiments not presented in this

chapter we permuted the linear system before the splitting and added correction terms to the splitted parts to ensure invertibility and numerical stability. This did not always seem to be a viable alternative, since it destroyed the structure of the original problem and generally slowed down the speed of convergence. For similar conclusions in the context of the preconditioned conjugate gradient method, see [38].

Due to its robustness, the restarted TSMRES method may be an attractive alternative to the restarted GMRES method when the linear system either readily has or can simply be preconditioned to have the form (2.1).

Chapter 6

Splittings and k-Hessenberg systems

6.1 Introduction

In this chapter, we consider solving the linear system

$$Hx = b, \tag{6.1}$$

where $H \in \mathbb{C}^{n\times n}$ is an upper (or a lower) k-Hessenberg matrix with density τ and $b \in \mathbb{C}^n$. By an upper k-Hessenberg matrix we mean an upper triangular matrix with k extra non-zero diagonals below the main diagonal, i.e., $h_{i,j} = 0$, $i > j + k$ and by the density $\tau = \tau(n)$ the ratio of the number of nonzero entries and n^2. When H is very sparse, τ is of the order $O(C/n)$ with some small constant C.

Linear systems of type (6.1) arise in modelling stochastic problems with Markov-chains and in the application of difference equations to the computation of some special functions. With these applications in mind, alternative methods to using the Gaussian elimination for solving block Hessenberg linear systems have been devised in [45, 50, 113].

Our interest in k-Hessenberg matrices originates from splittings for TSMRES as discussed in Chapter 5. Then, by first splitting a general matrix $A = L + R$, where L^T and R are upper k-Hessenberg matrices and then preconditioning by L from the left, for instance, we have the structure $I + S$ to which TSMRES can be applied. We also present how splittings involving k-Hessenberg matrices can be also used to extend the Gauss-Seidel iterations in general.

We propose a direct method based on the Gaussian elimination, without and with pivoting, for solving the linear system (6.1) and discuss the associated ILU preconditioning strategies. We then show that our method has a

computational complexity of the order $O(\tau(k+1)n^2)$ without pivoting. The method is also matrix-free in the sense that it does not rely on factoring the matrix explicitly and only requires knowing the entries of H locally. Considering storage, in addition to the storage needs for H, the method consumes a fixed amount of memory proportional to $O(kn)$. In contrast, even for sparse k-Hessenberg matrices the standard Gaussian elimination typically requires at least $O(n^2)$ flops together with the fact that the storage requirements can be prohibitive.

After analyzing the computational complexity, we consider numerical stability. The partial pivoting we propose is performed columnwise based on choosing, at each step, an unknown corresponding to the largest coefficient in modulus. Our analysis and numerical experiments indicate that this strategy is sufficient in most cases. Also threshold pivoting is considered, leading to significant reductions in complexity in most of the practical problems we have encountered.

These ideas lead naturally to a new factorization analogous to the LU-decomposition. We consider factoring a nonsingular matrix $A \in \mathbb{C}^{n\times n}$ into the product of a lower-triangular and a k-Hessenberg matrix. By applying the algorithms derived, linear systems involving A can be solved with this factorization similarly to the LU-decomposition. This is of particular interest in view of large scale problems since, for iterative methods such as GMRES, any factorization provides preconditioning opportunities once computed incompletely. Since the fill-in of the factors is different from the fill-in of the LU factors, new preconditioning opportunities of ILU-type arise. We regard k as a new parameter to be tuned to improve the standard ILU.

Finally, we illustrate the complexity and stability of our method for solving k-Hessenberg systems by numerical examples. We also consider the method as a preconditioner for an iterative method when used in conjunction with an incomplete decomposition of a matrix into a lower triangular and a k-Hessenberg matrix.

6.2 Solving sparse k-Hessenberg linear systems

In this section, we present a method for solving sparse k-Hessenberg linear systems efficiently. Approaches both without and with pivoting are considered. In the following, we describe the idea of the method in short.

In our approach to solving (6.1) without pivoting we start from the nth row and solve the unknown x_{n-k} in terms of the other k unknowns. We then

move upwards to the $(n-1)$th row and substitute the unknown just solved and solve x_{n-k-1} in terms of the same unknowns as x_{n-k}. Repeating this until up to the $(k+2)$th row, the representations for the $n-k-1$ unknowns are then substituted into to the first $k+1$ rows resulting in a $(k+1)$-by-$(k+1)$ linear system. Solving this with a direct method yields the values of the k unknowns. Using the representations computed yields the remaining $n-k-1$ unknowns and hence the solution to (6.1).

In our approach to solving (6.1) with pivoting we start from the nth row and solve one of the appearing $k+1$ unknowns in terms of the other k unknowns. An unknown with the largest coefficient in modulus is chosen as a pivot. We then move upwards to the $(n-1)$th row and substitute the unknown just solved. After finding the largest coefficients in modulus, a new pivot is chosen and the respective unknown is solved in terms of the other appearing k unknowns. Repeating this up to the $(k+2)$th row, the representations for the $n-k-1$ unknowns are then substituted into the first $k+1$ rows resulting in a $(k+1)$-by-$(k+1)$ linear system. Solving this with a direct method yields the values of the chosen k unknowns. The remaining $n-k-1$ unknowns and hence the solution to (6.1) is obtained by inserting these values into their representations. In other words, we have devised a generalized back substitution scheme with partial pivoting.

Consider the linear system (6.1) written componentwise as

$$\begin{pmatrix} h_{1,1} & h_{1,2} & & \cdots & & h_{1,n} \\ \vdots & \vdots & & \cdots & & \vdots \\ h_{k,1} & h_{k,2} & & \cdots & & h_{k,n} \\ 0 & h_{k+1,2} & & \cdots & & h_{k+1,n} \\ \vdots & \ddots & \ddots & \vdots & & \vdots \\ 0 & \cdots & 0 & h_{n,n-k} & \cdots & h_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_k \\ x_{k+1} \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_k \\ b_{k+1} \\ \vdots \\ b_n \end{pmatrix} \tag{6.2}$$

with $h_{i,j}, x_i, b_i \in \mathbb{C}$, for $1 \leq i, j \leq n$. We write the ith row of this system as

$$\sum_{j=\max\{1,i-k\}}^{n} h_{i,j}x_j = h^{(i)} \cdot x = b_i, \tag{6.3}$$

where $x \in \mathbb{C}^n$ and $h^{(i)} \in \mathbb{C}^n$ is the ith row vector of H. Denote the jth component of $h^{(i)}$ by $h^{(i)}_j$. Let

$$h^{(i)\setminus j} = \begin{pmatrix} 0 & \ldots & 0 & h_{i,i-k} & \ldots & h_{i,j-1} & 0 & h_{i,j+1} & \ldots & h_{i,n} \end{pmatrix}$$

be the row vector $h^{(i)}$ with its jth component replaced by zero.

6.2.1 A generalized back substitution scheme

Consider now solving (6.1) with a method that uses the steps of back substitution. For this, from nth row we have

$$h^{(n)} \cdot x = b_n. \tag{6.4}$$

Taking the row vector $h^{(n)}$, choose r_n to be the index satisfying

$$r_n = \min\{i \in \mathbb{N} \mid h_i^{(n)} \neq 0\}, \tag{6.5}$$

i.e., choose the index corresponding to the leftmost nonzero coefficient. Using $h_{r_n}^{(n)}$ as a pivot, solve from (6.4) the unknown x_{r_n} in terms of the other unknowns as

$$x_{r_n} = \frac{1}{h_{r_n}^{(n)}}(b_n - h^{(n)\setminus r_n} \cdot x). \tag{6.6}$$

After perfoming the divisions we have

$$x_{r_n} = \hat{b}_n - \hat{h}^{(n)\setminus r_n} \cdot x, \tag{6.7}$$

such that $\hat{b}_n$ and $\hat{h}^{(n)\setminus r_n}$ are stored in the memory. Moving on to the $(n-1)$th row, we substitute x_{r_n} into

$$h^{(n-1)} \cdot x = b_{n-1} \tag{6.8}$$

and compute the resulting coefficients. After the substitution, the coefficients in the $(n-1)$th row are

$$\tilde{h}^{(n-1)} = h^{(n-1)\setminus r_n} - h_{r_n}^{(n-1)} h^{(n)\setminus r_n}.$$

With these, we choose r_{n-1} analogously to (6.5) by setting

$$r_{n-1} = \min\{i \in \mathbb{N} \mid \tilde{h}_i^{(n-1)} \neq 0\}. \tag{6.9}$$

Using the coefficient $\tilde{h}_{r_{n-1}}^{(n-1)}$ as a pivot, solve $x_{r_{n-1}}$ in terms of the remaining unknowns as

$$x_{r_{n-1}} = \hat{b}_{n-1} + \hat{h}^{(n-1)\setminus r_{n-1}} \cdot x, \tag{6.10}$$

where

$$\hat{b}_{n-1} = \frac{1}{\tilde{h}_{r_{n-1}}^{(n-1)}}(b_{n-1} - h_{r_n}^{(n-1)} \hat{b}_n) \quad \text{and} \quad \hat{h}^{(n-1)\setminus r_{n-1}} = \frac{1}{\tilde{h}_{r_{n-1}}^{(n-1)}} \tilde{h}^{(n-1)\setminus r_{n-1}}$$

are again stored in memory. After this we move on the $(n-2)$th row, where the expressions (6.6) and (6.10) are used to solve the unknown $x_{r_{n-2}}$ in terms of other unknowns. The index r_{n-2} is chosen analogously to (6.5) and (6.9).

Proceeding in this manner, the modified coefficient vector of the jth row is computed by substituting the expressions for $x_n, x_{n-1}, \ldots, x_{j+1}$ recursively. Denoting by $h^{(t_i)}$ the modified row before the substitutions corresponding to the rows $i = n, \ldots, j+1$, the substitution procedure may be written as

$$\begin{aligned} h^{(t_n)} &= h^{(j)} \\ h^{(t_{i-1})} &= h^{(t_i)\setminus r_i} - h^{(t_i)}_{r_i} \hat{h}^{(i)\setminus r_i},\ i = n, \ldots, j+1. \end{aligned} \tag{6.11}$$

After completing (6.11), we have the relation $\tilde{h}^{(j)} = h^{(t_j)}$ for the modified coefficients of the jth row. The modified right-hand side is computed analogously. From $\tilde{h}^{(j)}$ we choose the index r_j as

$$r_j = \min\{i \in \mathbb{N} \mid \tilde{h}^{(j)}_i \neq 0\}, \tag{6.12}$$

and construct the expression for the unknown x_j as before.

Once the procedure reaches the $(k+2)$th row, the representations for the unknowns are substituted into the first $k+1$ rows to have a linear system

$$\tilde{A}\tilde{x} = \tilde{b} \tag{6.13}$$

with a coefficient matrix $\tilde{A} \in \mathbb{C}^{(k+1)\times(k+1)}$ and $\tilde{b} \in \mathbb{C}^{k+1}$. This linear system involves the unknowns corresponding to the indices $\mathcal{R} = \{i\}_{i=1}^{n} \setminus \{r_l\}_{l=k+2}^{n}$. Solving this small linear system using standard direct techniques yields the components x_i for $i \in \mathcal{R}$. The other components of x are then computed using $\hat{h}^{(j)\setminus r_j}$ and forward substitution starting from the $(k+2)$th row and moving down to the nth row. As a result this, we have Algorithm 6.1.

When constructing the representations for the unknowns x_j in the substitution phase of Algorithm 6.1, there is a division by $\tilde{h}^{(j)}_{r_j}$ for each $j = k+1, \ldots, n$. If this is smaller in modulus than some other components of $\tilde{h}^{(j)}$, it is possible that after a number of substitutions some coefficients become large, possibly leading to problems in numerical stability. If all the pivots are dominant in the columns $1 \leq i \leq n-k$ in the following sense, the computation of the representations for the unknowns in Algorithm 6.1 is numerically stable.

Definition 6.1. *A k-Hessenberg matrix $H \in \mathbb{C}^{n\times n}$ is subdiagonally dominant, if for each column $j = 1, \ldots, n-k$ it holds*

$$|h_{j+k,j}| \geq \sum_{i \neq j+k} |h_{i,j}|. \tag{6.14}$$

The stability of Algorithm 6.1 for subdiagonally dominant matrices is shown in section 6.4. The section also discusses ways to improve the stability by using threshold pivoting.

Algorithm 6.1 Algorithm for solving $Hx = b$

Require: $H \in \mathbb{C}^{n\times n}$ is an upper k-Hessenberg matrix, $k \geq 1$, $b \in \mathbb{C}^n$

for $i = n, \ldots, (k+2)$ **do**
  $h^{(t_n)} = H_i$, $\tilde{\tilde{b}} = b_i$
  for $j = n, \ldots, i+1$ **do**
    $h^{(t_{j-1})} = h^{(t_j)\setminus r_j} - h_{r_j}^{(t_j)} \hat{h}^{(j)\setminus r_j}$
    $\tilde{\tilde{b}} = \tilde{\tilde{b}} - h_{r_j}^{(t_j)} \hat{b}_j$
  end for
  Choose index r_i as in (6.5)
  $\hat{h}^{(i)\setminus r_i} = h^{(t_i)\setminus r_i} / h_{r_i}^{(t_i)}$, $\hat{b}_i = \tilde{\tilde{b}} / h_{r_i}^{(t_i)}$
end for
$\mathcal{R} = \{l\}_{l=1}^{n} \setminus \{r_l\}_{l=k+2}^{n}$
for $i = 1, \ldots, (k+1)$ **do**
  $h^{(t_n)} = H_i$, $\tilde{\tilde{b}} = b_i$
  for $j = n, \ldots, i+1$ **do**
    $h^{(t_{j-1})} = h^{(t_j)\setminus r_j} - h_{r_j}^{(t_j)} \hat{h}^{(j)\setminus r_j}$
    $\tilde{\tilde{b}} = \tilde{\tilde{b}} - h_{r_j}^{(t_j)} \hat{b}_j$
  end for
  Set ith row of $\tilde{A}$ to nonzero components of $h^{(t_i)}$
  Set ith component of $\tilde{b}$ to $\tilde{\tilde{b}}$
end for
Solve $\tilde{A}\tilde{x} = \tilde{b}$
for $i = k+2, \ldots, n$ **do**
  $x_{r_i} = \hat{b}_i - \hat{h}^{(i)\setminus r_i} \cdot x$
end for

Let us now describe how to partially pivot the algorithm. Starting from the nth row, choose the index r_n as

$$r_n = \{i \in \mathbb{N} \mid |h_i^{(n)}| \geq |h_j^{(n)}|,\ n-k \leq j \leq n\}, \tag{6.15}$$

i.e., r_n is chosen such that the coefficient $h_{r_n}^{(n)}$ is largest in modulus. As in (6.6), we solve x_{r_n} in terms of the other unknowns, giving

$$x_{r_n} = \hat{b}_n - \hat{h}^{(n)\setminus r_n} \cdot x, \tag{6.16}$$

where $\hat{b}_n$ and $\hat{h}^{(n)\setminus r_n}$ are of the same form as before, but possibly with a different pivot $h_{r_n}^{(n)}$ corresponding to r_n. From the construction we now have $|\hat{h}_j^{(n)\setminus r_n}| \leq 1$, for $n-k \leq j \leq n$.

As before, we substitute x_{r_n} into (6.8) and compute the resulting coefficient vector $\tilde{h}^{(n-1)}$. Choosing r_{n-1} as in (6.15) and dividing by the pivot $\tilde{h}^{(n-1)}_{r_{n-1}}$, we obtain the representation for $x_{r_{n-1}}$ as

$$x_{r_{n-1}} = \hat{b}_{n-1} + \hat{h}^{(n-1)\setminus r_{n-1}} \cdot x_p, \tag{6.17}$$

where $\hat{b}_{n-1}$ and $\hat{h}^{(n-1)\setminus r_{n-1}}$ are similar to (6.10), but possibly with a different divisor $\tilde{h}^{(n-1)}_{r_{n-1}}$.

We continue the substitutions recursively as before up to the $(k+2)$th row. During the substitutions it is possible that coefficients which were originally zero become nonzero and require additional substitutions compared to the scheme without pivoting, i.e., $h^{(t_{i-1})}$ in (6.11) can contain nonzero coefficients which were zero in the original $h^{(j)}$ and require substitutions. When the substitutions have been performed, we choose on the jth row the pivot r_j as

$$r_j = \{i \in \mathbb{N} \mid |\tilde{h}^{(j)}_i| \geq |\tilde{h}^{(j)}_l|,\ j-k \leq l \leq n\}, \tag{6.18}$$

where $\tilde{h}^{(j)}_i = h^{(t_j)}$.

Once the substitutions are completed, the algorithm proceeds by solving the respective small linear system (6.13). The other unknowns of x are then obtained by forward substitution.

The difference between the non-pivoted and pivoted versions lies in the choice of the indices r_j. In Algorithm 6.1 this appears on line 7 which is replaced with (6.18) in the pivoted version. In exact arithmetic the two versions of the algorithm produce the same solution. In finite precision arithmetic, the pivoted version is numerically more stable, since the modified coefficients satisfy $|\hat{h}^{(j)\setminus r_j}_i| \leq 1$, for $(j-k) \leq i \leq n$.

6.2.2 Interpretation by using row and column operations

Next we present Algorithm 6.1 for solving (6.1) by using elementary row and column operations. Then Algorithm 6.1 can be viewed as a transformation of the linear system $Hx = b$ into an equivalent form $L\tilde{x} = \tilde{b}$ with permuted unknowns $\tilde{x} = \Pi x$. The matrix L is diagonal except that its $(k+1)$ first columns are possibly nonzero. The transformed linear system is readily solvable and once $\tilde{x}$ is computed we have $x = \Pi^T \tilde{x}$.

Define $\Pi_{i,j}$ to be an elementary permutation matrix that swaps the ith and jth columns when operated from the right. For short, denote $\Pi_n = \Pi_{r_n,n}$, where r_n is chosen as in (6.5). Operating with Π_n from the right yields

$$H\Pi_n\tilde{x} = \tilde{H}\tilde{x} = b, \text{ with } \tilde{x} = \Pi_n x. \tag{6.19}$$

Choose a Gauss transformation $M_n = I - t_n e_n^T$ with

$$t_n = \frac{1}{h_{n,r_n}} \left(h_{1,r_n} \ \cdots \ h_{n-1,r_n} \ 0 \right).$$

Applying M_n to (6.19) from the left yields

$$H_1 \tilde{x} = \hat{b}, \tag{6.20}$$

where $\hat{b} = M_n b$ and $H_1 = M_n \tilde{H}$. Assuming, for simplicity, $r_n = n - k$, the matrix H_1 has the structure

$$H_1 = \begin{pmatrix} h_{1,1} & \cdots & h_{1,n-k-1} & \tilde{h}_{1,n} & \cdots & \tilde{h}_{1,n-1} & 0 \\ \vdots & & \vdots & \vdots & & \vdots & \vdots \\ h_{k+1,1} & \cdots & h_{k+1,n} & \tilde{h}_{k+1,n} & \cdots & \tilde{h}_{k+1,n-1} & 0 \\ 0 & \ddots & & \vdots & & \vdots & \vdots \\ \vdots & & h_{n-1,n-k-1} & \tilde{h}_{n-1,n} & \cdots & \tilde{h}_{n-1,n-1} & 0 \\ 0 & \cdots & 0 & h_{n,n} & \cdots & h_{n,n-1} & h_{n,r_n} \end{pmatrix}, \tag{6.21}$$

where only the entries $\tilde{h}_{i,j}$ differ from those of $\tilde{H}$. We write (6.21) blockwise as

$$H_1 = \begin{pmatrix} H_{n-1} & h_{n-k-1} & \tilde{H}_{n-1} & \tilde{h}_{n-1} & \mathbf{0} \\ \mathbf{0} & 0 & \hat{h}_n & h_{n,n-1} & h_{n,n} \end{pmatrix}, \tag{6.22}$$

where $H_{n-1} \in \mathbb{C}^{(n-1)\times(n-k-2)}$, $\tilde{H}_{n-1} \in \mathbb{C}^{(n-1)\times(k-1)}$, $\hat{h}_n \in \mathbb{C}^{k-1}$, h_{n-k-1}, $\tilde{h}_{n-1} \in \mathbb{C}^{n-1}$ and all-zero matrices $\mathbf{0}$ are of appropriate size.

From (6.22) we observe that the $(n-k-1)$th column has remained unmodified in the application by M_n. Operating from the right with the permutation $\Pi_{n-1} = \Pi_{r_{n-1},n-1}$ yields

$$H_1 \Pi_{n-1} \tilde{x} = \tilde{H}_1 \tilde{x} = \hat{b}, \text{ with } \tilde{x} = \Pi_{n-1} \Pi_n x, \tag{6.23}$$

where $\tilde{H}_1$, again assuming $r_{n-1} = n - k - 1$, has the structure

$$\tilde{H}_1 = \begin{pmatrix} H_{n-1} & \tilde{h}_{n-1} & \tilde{H}_{n-1} & h_{n-k-1} & \mathbf{0} \\ \mathbf{0} & h_{n,n-1} & \hat{h}_n & 0 & h_{n,n} \end{pmatrix}. \tag{6.24}$$

Now the elimination is carried out with the $(n-k-1)$th column by choosing a Gauss transformation $M_{n-1} = I - t_{n-1} e_{n-1}^T$ with

$$t_{n-1} = \frac{1}{h_{n-1,n-k-1}} \left(h_{1,n-k-1} \ \cdots \ h_{n-1,n-k-1} \ 0 \ 0 \right).$$

With this, the second elimination step yields

$$M_{n-1}\tilde{H}_1\tilde{x} = H_2\tilde{x} = M_{n-1}\hat{b} = \hat{\tilde{b}}. \tag{6.25}$$

When its last row and column have been removed, H_2 can be partitioned as H_1 in (6.22).

Applying the Gauss transformations M_j, for $j = n, \ldots, k+2$, in this manner, transforms the original linear system into

$$L\tilde{x} = \tilde{b}, \tag{6.26}$$

where $\tilde{x} = \Pi_1 \cdots \Pi_{n-2}\Pi_{n-1}\Pi_n x = \Pi x$ and L is diagonal except for its first k columns and a $(k+1)$-by-$(k+1)$ block in the upper left corner. Once the unknowns corresponding to the $(k+1)$-by-$(k+1)$ block have been solved, the remaining unknowns are obtained with forward substitution. The original unknowns are found by setting $x = \Pi^T\tilde{x}$.

Let us illustrate these steps with an example.

Example 6.1 Consider a k-Hessenberg matrix with $k = 1$ and $n = 5$. Then L has a 2-by-2 block in the upper left corner and is otherwise diagonal, except for the first column. Denoting a nonzero entry by x and modified entry by x̃, the elimination proceeds as presented in Figure 6.1.

$$\begin{pmatrix} \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & 0 & 0 & \mathrm{x} & \mathrm{x} \end{pmatrix} \xrightarrow{M_5\Pi_4\Pi_5} \begin{pmatrix} \mathrm{x} & \mathrm{x} & \tilde{\mathrm{x}} & \mathrm{x} & 0 \\ \mathrm{x} & \mathrm{x} & \tilde{\mathrm{x}} & \mathrm{x} & 0 \\ 0 & \mathrm{x} & \tilde{\mathrm{x}} & \mathrm{x} & 0 \\ 0 & 0 & \tilde{\mathrm{x}} & \mathrm{x} & 0 \\ 0 & 0 & \mathrm{x} & 0 & \mathrm{x} \end{pmatrix} \rightarrow \cdots \rightarrow \begin{pmatrix} \tilde{\mathrm{x}} & \mathrm{x} & 0 & 0 & 0 \\ \tilde{\mathrm{x}} & \mathrm{x} & 0 & 0 & 0 \\ \tilde{\mathrm{x}} & 0 & \mathrm{x} & 0 & 0 \\ \tilde{\mathrm{x}} & 0 & 0 & \mathrm{x} & 0 \\ \tilde{\mathrm{x}} & 0 & 0 & 0 & \mathrm{x} \end{pmatrix}$$

Figure 6.1: Elimination procedure for $n \times n$ Hessenberg matrix.

Since the pivots $h_{r_j}^{(j)}$ in Algorithm 6.1 and in this implementation are equal for rows from n to $k+2$, the coefficients $\hat{h}^{(j)\backslash r_j}$ are also equal. By performing the substitutions into the first rows, the same small $(k+1)$-by-$(k+1)$ linear system is obtained. Thus the scheme presented in this section is equivalent to Algorithm 6.1.

6.3 Computational complexity

Algorithm 6.1 consists of three distinct phases. First the representations are built for $n-k-1$ different unknowns, after which the small $(k+1)$-by-$(k+1)$ linear system is constructed. In the final phase the small linear

system is solved and the solution of the original linear system is formed with forward substitutions. By analyzing and combining the complexities of these phases, we get the following proposition.

Proposition 6.2. *The computational complexity of Algorithm 6.1 with the pivot choice* (6.12) *for the linear system* (6.1) *involving an invertible k-Hessenberg matrix with density* τ *is of the order*

$$O(\tau(k+1)n^2). \tag{6.27}$$

In addition to storing the matrix, the storage required by the Algorithm 6.1 is

$$O(kn) \tag{6.28}$$

floating point numbers.

Proof. In the first phase, on lines from 1 to 9 of Algorithm 6.1, a loop runs over all the rows from n to $k+2$. For each row i, on lines from 3 to 6, a substitution is performed for those unknowns of the previous rows j, for $j = i+1, \ldots, n$, which have nonzero coefficients in $h^{(t_j)}$. One multiplication and addition with a vector with k nonzeroes at most is required to modify $h^{(t_j)}$ in a single substitution. In addition, the modification of $\tilde{\tilde{b}}$ requires 2 operations per substitution.

On line 7, the index r_i can be computed in $O(1)$ time, if the indices of the nonzero components of the sparse vector $\tilde{h}^{(j)}$ are ordered. At this point, $\tilde{h}^{(j)}$ has only $k+1$ nonzero components. Forming of the new unknowns $h^{(i)\setminus r_i}$ and $\hat{b}_i$ on the line 8 requires $k+1$ operations. Combining the complexities, we get the total number of operations for the first phase as

$$\begin{aligned} & 2k + \sum_{i=k+2}^{n-1} \left(1 + (k+1) + \tau \sum_{j=i+1}^{n} 2(k+1) \right) \\ = \;& O(\tau(k+1)n^2), \end{aligned} \tag{6.29}$$

since $\sum_{i=k+2}^{n-1} \sum_{j=i+1}^{n} 1 = n(n-k-2) - (\frac{1}{2}n^2 - \frac{1}{2}n - \frac{1}{2}k^2 - \frac{3}{2}k - 1)$.

The second phase consists of lines $10-19$. The construction of the set $\mathcal{R}$ requires n operations. The generation of the small $(k+1)$-by-$(k+1)$ linear system (6.13) requires a substitution of the unknowns generated in the first phase to the first $k+1$ rows. The substitution on lines $13-16$ is of same form as in the first phase of the algorithm. The operation count for the second phase thus is

$$\begin{aligned} n + \tau \sum_{i=1}^{k+1} \sum_{j=k+2}^{n} 2(k+1) &= 2\tau(k+1)^2(n-k-1) + n \\ &= O(\tau(k+1)^2 n + n). \end{aligned} \tag{6.30}$$

The third phase on lines $20 - 23$ begins with the solution of the full linear system (6.13). By using standard techniques, this system requires in the order of $O(k^3)$ operations to solve. Using the solved unknowns x_i, with $i \in \mathcal{R}$ and their representations $\hat{h}^{(i)\setminus r_i}$, for $i = k+2, \ldots, n$, the algorithm yields the solution to (6.1). Line 22 performs one inner product of a sparse vector of length n with $k+1$ nonzero entries and an addition with scalar. The combined complexity for these operations is

$$\begin{aligned} \sum_{i=k+2}^{n} (2k+1) + O(k^3) &= (2k+1)(n-k-1) + O(k^3) \qquad (6.31) \\ &= O((2k+1)n) + O(k^3) \end{aligned}$$

The complexity of the algorithm then follows after summing (6.29), (6.30) and (6.31).

To obtain the memory requirement, we combine the size of the coefficient matrix $\hat{h}^{(i)\setminus r_i}$, the modified right-hand side $\hat{b}$ and the reduced equation with matrix $\tilde{A}$ and vector $\tilde{b}$. Floating point storage requirement hence becomes

$$kn + n + k^2 + k = O(kn). \qquad (6.32)$$

□

When the matrix H is sparse and subdiagonally dominant, Algorithm 6.1 is computationally very efficient both in terms of memory and the number of floating point operations consumed. Note that in the applications we have in mind, $\tau = C/n$ with a small constant C. Then (6.27) reduces to $O(C(k+1)n)$.

Example 6.2 For a familiar example, consider the linear system

$$Bx = b, \qquad (6.33)$$

where $B \in \mathbb{C}^{n \times n}$ is a sparse band matrix with lower and upper bandwidths k_l and k_u. Without loss of generality we may assume that $k_u \geq k_l$. As we have at most k_u nonzeroes above the diagonal at each column, the density of the system can be written as $\tau = k_u/n$. Then Proposition 6.2 gives the complexity of Algorithm 6.1 for banded systems as $O((k_l+1)k_u n)$.

Proposition 6.2 is concerned with the unpivoted version of Algorithm 6.1. In the first phase of the partially pivoted version of Algorithm 6.1, the substitutions are made after inspecting the modulus of the nonzero components of the vector $h^{(t_j)}$ in the formula (6.11). Then the computational complexity (6.29) does not depend on the sparsity structure of the original matrix

directly. In the worst case the complexity can amount to be of the same order as that of solving a full k-Hessenberg system, which is $O((k+1)n^2)$. The same relation holds for the complexity of the second phase, which in the worst case is of the order $O((k+1)^2n+n)$. Combining this with (6.31), we get the worst case complexity of the partially pivoted version of Algorithm 6.1 to be

$$O((k+1)n^2) \tag{6.34}$$

which is the complexity of the Gaussian elimination with partial pivoting for k-Hessenberg systems. Threshold pivoting considered in the next section provides a way to decrease this.

6.4 Numerical stability and threshold pivoting

It is well known that when the standard Gaussian elimination is applied without pivoting, the elements of the transformed system can grow during the elimination procedure. Analogously, when Algorithm 6.1 is applied without pivoting, the choice of pivots may result in growth of the elements in the representations for the unknowns $\hat{h}^{(j)\setminus r_j}$. In both cases, the growth can lead to a loss of precision in the computed solution. Note that for an upper Hessenberg matrix with $k=1$, the standard Gaussian elimination is numerically backward stable, see [61, Theorem 9.10, pp. 172].

We now derive error bounds and give sufficient conditions for stability for Algorithm 6.1. As described in section 6.2.2, Algorithm 6.1 can be implemented in terms of standard row and column operations. This relation makes the error analysis quite similar to the error analysis of the standard Gaussian elimination. We only need to observe that Algorithm 6.1 actually solves a row and column permuted linear system

$$\tilde{H}\tilde{x} = \Pi_r H \Pi_c \tilde{x} = \Pi_r b = \tilde{b},\ \tilde{x} = \Pi_c x, \tag{6.35}$$

where

$$\tilde{H} = \begin{pmatrix} h_{n,n-k} & 0 & & \dots & h_{n,n} & \dots & h_{n,n-k+1} \\ h_{n-1,n-k} & h_{n-1,n-k-1} & 0 & \dots & h_{n-1,n} & \dots & h_{n-1,n-k+1} \\ \vdots & \ddots & \ddots & & \vdots & & \vdots \\ h_{k+1,n-k} & \dots & & h_{k+1,1} & h_{k+1,n} & \dots & h_{k+1,n-k+1} \\ \vdots & & & \vdots & \vdots & & \vdots \\ h_{1,n-k} & \dots & & h_{1,1} & h_{1,n} & \dots & h_{1,n-k+1} \end{pmatrix} \tag{6.36}$$

and Π_r and Π_c are suitably chosen permutations. By eliminating the strictly lower triangular part of the first $n-k$ columns, the linear system $\tilde{H}\tilde{x} = \tilde{b}$ is transformed into $U\tilde{x} = \tilde{\tilde{b}}$, with U diagonal except for the last k columns and a $(k+1)$-by-$(k+1)$ block in the lower right corner. In exact arithmetic, the transformation to the upper triangular form is equivalent to forming a factorized linear system $LU\tilde{x} = \tilde{b}$, with $LU = \tilde{H}$, where L is lower triangular except for the last $(k+1)$ columns.

In finite precision arithmetic a slightly perturbed system $\hat{L}\hat{U}\tilde{x} = \tilde{b}$ is solved instead. Higham [61] presents the error for the computed LU factors as

$$\hat{L}\hat{U} = \tilde{H} + \Delta\tilde{H},\ |\Delta\tilde{H}| \leq \gamma_n|\hat{L}||\hat{U}|,$$

where $\gamma_n = \frac{nu}{1-nu}$, u is the unit roundoff and the inequalities between matrices are assumed to hold componentwise. Backward error of the Gaussian elimination for the computed solution $\hat{x}$ of $\tilde{H}\tilde{x} = \tilde{b}$ is then

$$(\tilde{H} + \Delta\tilde{H})\hat{x} = \tilde{b},\ |\Delta\tilde{H}| \leq \gamma_{3n}|\hat{L}||\hat{U}|. \tag{6.37}$$

The ratio of the factors and the original matrix is not bounded without pivoting and we may have $|||\hat{L}||\hat{U}|||/||\tilde{H}||$ arbitrarily large. By inspecting our elimination procedure, we observe that the partially pivoted version of Algorithm 6.1 corresponds to the Gaussian elimination with column pivoting involving the matrix (6.36). From van Veldhuizen [124] we have

$$(\tilde{H} + \Delta\tilde{H})\hat{x} = \tilde{b},\ ||\tilde{H}|| \leq n^2\gamma_{3n}\rho_n||\tilde{H}||, \tag{6.38}$$

where ρ_n is the growth factor defined as

$$\rho_n = \frac{\max_{i,j,l}|h_{i,j}^{(l)}|}{\max_{i,j}|h_{i,j}|}, \tag{6.39}$$

where $h_{i,j}^{(l)}$, for $l = 1, \ldots, n$, are the elements that occur at the lth elimination step on the $(n-l)$th row.

If H is subdiagonally dominant, Algorithm 6.1 is numerically stable without pivoting as follows.

Proposition 6.3. *For a subdiagonally dominant k-Hessenberg matrix H, Algorithm 6.1 is numerically stable.*

Proof. By the subdiagonal dominance, for the permuted system (6.36) we have

$$|h_{j,j}| \geq \sum_{i \neq j}|h_{i,j}|, \quad \text{for } j = 1, \ldots, n-k, \tag{6.40}$$

i.e., the first $n-k$ columns in the permuted system are diagonally dominant. The first step of elimination can be written as

$$\begin{pmatrix} \alpha & \beta \\ \gamma & H_1 \end{pmatrix} = \begin{pmatrix} 1 & \mathbf{0} \\ \alpha^{-1}\gamma & I \end{pmatrix} \begin{pmatrix} \alpha & \beta \\ 0 & H^{(1)} \end{pmatrix},$$

where $\alpha \in \mathbb{R}$, $\beta, \gamma \in \mathbb{C}^{n-1}$ and $H^{(1)} = H_1 - \alpha^{-1}\gamma\beta \in \mathbb{C}^{(n-1)\times(n-1)}$. The subdiagonal dominance is preserved, if it is preserved in a single step of elimination. Following Higham [61, Theorem 13.7 and 13.8], for columns j from 1 to $n-k$, we have after a single step

$$\begin{aligned} \sum_{i=2,i\neq j} |h_{i,j}^{(1)}| &= \sum_{i=2,i\neq j} |h_{i,j} - \frac{\gamma_i\beta_j}{\alpha}| \\ &\leq \sum_{i=2,i\neq j} |h_{i,j}| + \frac{|\beta_j|}{\alpha}(|h_{1,1}| - |\gamma_j|) \quad \text{by (6.40)} \\ &\leq |h_{j,j}| - |\gamma_j|\frac{|\beta_j|}{\alpha} \quad \text{by (6.40)} \\ &= |h_{j,j}^{(1)}|, \end{aligned}$$

which implies the subdiagonal dominance of the matrix $H^{(1)}$.

From the subdiagonal dominance, it follows $|l_{i,j}| \leq 1$. For the intermediate steps $H^{(l)}$, we have for columns j from 1 to $n-k$

$$\begin{aligned} \sum_{i=l}^{n} |h_{i,j}^{(l)}| &= \sum_{i=l}^{n} |h_{i,j}^{(l-1)} - \frac{h_{i,(l-1)}^{(l-1)} h_{(l-1),j}^{(l-1)}}{h_{(l-1),(l-1)}^{(l-1)}}| \leq \sum_{i=l-1}^{n} |h_{i,j}^{(l-1)}| \\ &\leq \cdots \leq \sum_{i=1}^{n} |h_{i,j}|, \end{aligned}$$

by using the subdiagonal dominance. Now for the elements of U we have

$$\begin{aligned} \max_{l\leq i,j\leq n} |h_{i,j}^{(l)}| &\leq \max_{l\leq j\leq n} \sum_{i=l}^{n} |h_{i,j}^{(l)}| \leq \max_{l\leq j\leq n} \sum_{i=1}^{n} |h_{i,j}| \\ &\leq 2 \max_{1\leq j\leq n} |h_{j,j}|. \end{aligned}$$

Using this and (6.39), the growth factor for the representations of the unknowns in Algorithm 6.1 without pivoting to equal $\rho = 2$. This implies the numerical stability of Algorithm 6.1 for subdiagonally dominant matrices, as long as the small $(k+1)$-by-$(k+1)$ linear system is solved with the fully pivoted Gaussian elimination. □

For matrices which are not subdiagonally dominant, sparsity can be taken advantage of and satisfactory numerical accuracy is often obtained by using threshold pivoting. The purpose of threshold pivoting is to decrease the computational complexity at the cost of a minor decrease in numerical stability; see [36] for details. Choosing a threshold $\alpha = (0, 1]$, we select the pivot element at the lth elimination step among the set of elements satisfying

$$|h_{i,j}^{(l)}| \geq \alpha \max_m |h_{i,m}^{(l)}|. \tag{6.41}$$

For Algorithm 6.1 with threshold pivoting, the growth of the elements is bounded during the elimination as follows.

Proposition 6.4. *For one elimination step we have*

$$\max_j |h_{i,j}^{(l+1)}| \leq (1 + \alpha^{-1}) \max_j |h_{i,j}^{(l)}|.$$

The growth of the elements in the whole elimination procedure is bounded as

$$\max_j |h_{i,j}^{(l+1)}| \leq (1 + \alpha^{-1})^{p_j} \max_j |h_{i,j}^{(l)}|,$$

where p_j is the number of off-diagonal entries in the jth column of U.

Proof. In the elimination procedure, the intermediate steps l and $l + 1$ are related by the equation

$$h_{i,j}^{(l+1)} = h_{i,j}^{(l)} - \big(\frac{h_{i,l}^{(l)}}{h_{l,l}^{(l)}}\big) h_{l,j}^{(l)}, \ i, j > l,$$

from which we have

$$|h_{i,j}^{(l+1)}| \leq \max\{|h_{i,j}^{(l)}|, |h_{i,l}^{(l)}|\}\big(1 + \frac{|h_{l,j}^{(l)}|}{|h_{l,l}^{(l)}|}\big),$$

from which the bound for one step of elimination follows by taking the maximum over the columns j and using (6.41). Then we have

$$\max_j |h_{i,j}^{(l+1)}| \leq \max_j |h_{i,j}^{(l)}|\big(1 + \frac{\max_j |h_{l,j}^{(l)}|}{|h_{l,l}^{(l)}|}\big) \leq \big(1 + \frac{1}{\alpha}\big) \max_j |h_{i,j}^{(l)}|.$$

The bound for the whole elimination follows, since the jth column of U is changed at most p_j times during the elimination. □

This gives us the following bound on the growth factor.

Proposition 6.5. *The growth factor for Algorithm 6.1 with threshold pivoting is*

$$\rho \leq (1 + \alpha^{-1})^{p_j}.$$

Proof. The proof follows directly by using the Proposition 6.4 and the definition of the growth factor (6.39). □

Our numerical experiments indicate that the bound obtained in Proposition 6.5 for threshold pivoted Algorithm 6.1 is somewhat pessimistic. In many cases it appears to suffice to choose a very small threshold α for a reasonable accuracy. This may well be due to the fact that the numerical fill-in of the system remains reasonably small since the leftmost nonzero coefficients in the modified rows are typically chosen as pivots.

6.5 Triangular-Hessenberg decompositions

In what follows, a factorization of a matrix $A \in \mathbb{C}^{n\times n}$ into the product of a triangular matrix and a k-Hessenberg matrix is derived. Since the complexity of Algorithm 6.1 is of the order $O(\tau(k+1)n^2)$ for k-Hessenberg systems, a decomposition

$$\Pi A = LH, \tag{6.42}$$

can be used to solve linear systems involving A analogously to the LU-decomposition. Here H is an upper k-Hessenberg, L is a lower triangular and Π a permutation matrix resulting from partial pivoting.

Another application of this decomposition lies in preconditioning sparse linear systems. In an incomplete LH-decomposition we compute $A \approx \hat{L}\hat{H}$ in such a way that $\hat{L}$ and $\hat{H}$ remain sparse. Since the complexity of Algorithm 6.1 for solving sparse linear systems with density $\tau = C/n$ is $O(C(k+1)n)$, this approximate factorization of A can be used as a preconditioner for an iterative method analogously to the ILU preconditioning approach [103, 105, 85, 129].

To compute the decomposition (6.42), start from the $(k+1)$th row of the matrix A by finding an element $a_{l,1}$ satisfying

$$|a_{l,1}| = \max_{k+1\leq j\leq n} \{|a_{j,1}|\}, \tag{6.43}$$

i.e., $a_{l,1}$ is the largest element in modulus in the first column of A below the kth entry. Let Π_1 be a permutation matrix which swaps the $(k+1)$th and the lth row when operated from the left. Choose a Gauss transformation

$M_1 = I - t_1 e_1^T$ with $t_1 = \begin{pmatrix} 0 & \dots & 0 & \frac{a_{k+2,1}}{a_{l,1}} & \dots & \frac{a_{n,1}}{a_{l,1}} \end{pmatrix}$ to have $M_1 \Pi_1 A = A_1$, where

$$A_1 = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & & & \\ a_{l,1} & a_{l,2} & \cdots & a_{l,n} \\ 0 & & & \\ \vdots & & \widetilde{A} & \\ 0 & & & \end{pmatrix}. \tag{6.44}$$

Continuing the process recursively with the submatrix $\widetilde{A}$ yields

$$M_{n-k-1} \Pi_{n-k-1} \cdots M_1 \Pi_1 A = H, \tag{6.45}$$

with a k-Hessenberg matrix H. Since $e_l^T \Pi_j = e_l^T$ for $j > l$, we have

$$\Pi_j M_l = \Pi_j - \Pi_j t_l^T e_l^T \Pi_j = (I - \Pi_j t_l^T e_l) \Pi_j. \tag{6.46}$$

By denoting $\tilde{t}_l = \Pi_{n-k-1} \cdots \Pi_{l+1} t_l$ and $\widetilde{M}_l = I - \tilde{t}_l^T e_l$ yields

$$\begin{aligned} H &= M_{n-k-1} \Pi_{n-k-1} M_{n-k-2} \Pi_{n-k-2} \cdots M_1 \Pi_1 A \\ &= \widetilde{M}_{n-k-1} \widetilde{M}_{n-k-2} \Pi_{n-k-1} \Pi_{n-k-2} \cdots M_1 \Pi_1 A \\ &= \widetilde{M}_{n-k-1} \widetilde{M}_{n-k-2} \widetilde{M}_{n-k-3} \Pi_{n-k-1} \Pi_{n-k-2} \Pi_{n-k-3} M_{n-k-4} \cdots M_1 \Pi_1 A \\ &= \vdots \\ &= \widetilde{M}_{n-k-1} \widetilde{M}_{n-k-2} \cdots \widetilde{M}_1 \Pi_{n-k-1} \cdots \Pi_1 A. \end{aligned}$$

It remains to recover the lower triangular part of the LH-decomposition from $L = (\widetilde{M}_{n-k-1} \cdots \widetilde{M}_2 \widetilde{M}_1)^{-1}$. Since $\widetilde{M}_j^{-1} = I + \tilde{t}_j e_j^T$ and $e_j^T \tilde{t}_l = 0$ for $j \le l$, we have

$$L = \widetilde{M}_1^{-1} \cdots \widetilde{M}_{n-k-1}^{-1} = I + \sum_{j=1}^{n-k-1} \tilde{t}_j e_j^T \tag{6.47}$$

and a permutation $\Pi = \Pi_{n-k-1} \cdots \Pi_1$. This completes the computation of the decomposition (6.42). Note that the first k columns of L, except for the ones on the diagonal, are zeros.

When the kth subdiagonal of A is dominant, i.e., $|a_{j+k,j}| \ge \sum_{i \ne j+k} |a_{i,j}|$ holds for $j = 1, \dots, n-k$, in the resulting decomposition H is subdiagonally dominant. We then have the following proposition.

Proposition 6.6. *Suppose the kth subdiagonal of $A \in \mathbb{C}^{n \times n}$ is dominant. Then the elimination produces a decomposition $\Pi A = LH$ such that $|l_{i,j}| \le 1$ and H is subdiagonally dominant k-Hessenberg matrix.*

Proof. From the structure of A it is clear that $|l_{i,j}| \leq 1$ results. By a similar argument as in the proof of Proposition 6.3, the subdiagonal dominance is preserved in the submatrix $\widetilde{A}$. The claim then follows by induction. □

Algorithm 6.1 is reliable when H is subdiagonally dominant. Hence when the kth subdiagonal of A is dominant, the LH-decomposition has many of the same qualities as the LU-decomposition by the fact that solving with the k-Hessenberg factor does not require partial pivoting. Observe that for a diagonally dominant A the dominance of the kth subdiagonal can be organized by permuting the main diagonal down by k diagonals by operating from the left with the permutation

$$Q = \begin{pmatrix} \mathbf{0} & I_k \\ I_{n-k} & \mathbf{0} \end{pmatrix}, \tag{6.48}$$

where I_j is an j-by-j identity matrix of and $\mathbf{0}$ denotes a zero matrix of appropriate size. Therefore the factorization can be used in many relevant applications.

Since the first k rows of A remain unmodified in the decomposition process, possibilities for minimizing the fill-in and the computational cost of the LH-decomposition arise. For example, we may choose an initial permutation Q in such a way that those k rows of A which contain the most entries are shifted to be the first k rows of QA.

When an LH-decomposition is constructed in an incomplete fashion, the usual techniques for dropping small coefficients in $\tilde{A}$ can be used to enforce sparsity of the factors. By using a suitable downward shift (6.48), we can guarantee the existence of LH decomposition for diagonally dominant matrices A.

Denoting by a_i, l_i and h_i the ith row and by $a_{i,j}$, $l_{i,j}$ and $h_{i,j}$ element (i,j) of matrices A, L and H, we have Algorithm 6.2 for incomplete LH-decomposition. Algorithm 6.2 is based on the IKJ ordering of the Gaussian elimination. On lines 7 and 12, a suitable dropping stategy must be chosen to maintain the sparsity in the computed factors. The dropping can be based on the numerical properties of w or the sparsity structure of the original matrix. Unfortunately the subject of choosing a suitable dropping strategy is too broad to be discussed here extensively. For different types of incomplete factorization based preconditioning techniques and numerical dropping strategies, see for example [7, 105] and references therein.

Algorithm 6.2 Incomplete LH-decomposition $A \approx LH$

Set $h_i = a_i$ and $l_i = e_i^T$, $i = 1, \ldots, k$
for $i = k+1, \ldots, n$ **do**
    $w = a_i$
    for $l = k+1, \ldots, i-1$ **do**
        $c = l - k$
        $w_c = w_c / h_{l,c}$
        Apply numerical dropping to w
        if $w_c \neq 0$ **then**
            $w = w - w_c h_l$
        end if
    end for
    Apply dropping to w
    $l_{i,j} = w_j$, $j = 1, \ldots, i-k-1$, $l_{i,i} = 1$
    $h_{i,j} = w_j$, $j = i-k, \ldots, n$
end for

6.6 Numerical experiments

Next we present numerical experiments to illustrate the properties of Algorithm 6.1 in practice. All the tests were performed by using MATLAB [84], whose syntax we use.

In the first numerical example, we illustrate the computational complexity and the effect of threshold pivoting versus the dimension of the problem.

Example 6.3 We generate a sparse matrix $A \in \mathbb{R}^{n \times n}$ with normally distributed entries and an average density $\tau = 0.1$ by using the `sprandn`-function and then extracting its upper k-Hessenberg part with the `triu`-function, i.e., we set

$$A = \text{triu}(\text{sprandn}(n, n, \tau), -k). \tag{6.49}$$

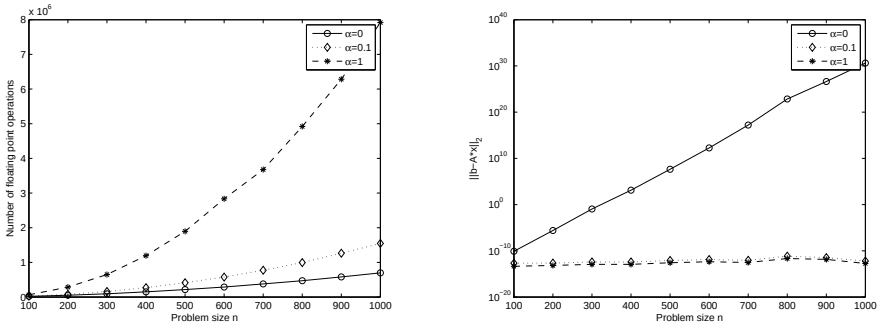
In addition, we generate a band matrix B with bandwidth k by setting

$$B = kD_0 + \sum_{j=k-2}^{k} j(D_j + D_j^T), \tag{6.50}$$

where $D_j =$`spdiags`$($`ones`$(n, 1), j, n, n)$, $j \geq 0$. With these, the upper k-Hessenberg matrix is defined by

$$H = A + B. \tag{6.51}$$

Set $k = 10$ and $n = 100j$, $j = 1, \ldots, 10$ and solve $Hx = b$ with a randomly generated b by using Algorithm 6.1 with three different pivoting thresholds. The results of computations are depicted in Figure 6.2 illustrating the number of floating point operations and the norms of the residuals $||b - Hx||_2$ of the solution x versus the problem dimension n. The pivoting thresholds were $\alpha = 0, 0.1$ and 1.



(a) Floating point operations (b) The norm $||b - Hx||_2$

Figure 6.2: Results of Example 6.3 versus the dimension n

When no pivoting is used, the $O(C(k+1)n)$ complexity of Algorithm 6.1 can be observed in Figure 6.2. As predicted by the bound (6.38), in Figure 6.2(b) we observe that the error of the computation is not bounded without pivoting as the dimension grows. However, a very modest use of threshold pivoting is enough to ensure satisfactory numerical stability and a nearly linear operation count.

We now give a more realistic example.

Example 6.4 We take A to be the matrix **pde900** from the Matrix Market collection [92]. It arises from a finite difference discretization of a two-dimensional variable-coefficient linear elliptic equation. The matrix is real, banded and unsymmetric with $n = 900$ having 4380 nonzero entries and a lower bandwidth $k = 29$. The condition number estimate for A is $2.9E+02$.

We again count the number of floating point operations and compute the residuals for solving $Ax = b$ with a right-hand side $b = \hat{b}/||\hat{b}||_2$ where $\hat{b}$ has normally distributed, randomly generated entries. The results obtained with Algorithm 6.1 with $k = 29$ by using various thresholds α are presented in Table 6.1.

The results are much like those of Example 6.3. We can clearly see how already a tiny threshold improved the numerical stability of Algorithm 6.1

α	#flops	$\|\|b - Ax\|\|_2/\|\|b\|\|_2$
0	254633	$4.3711e+03$
$1e-4$	527890	$2.0897e-12$
0.1	1092674	$3.9854e-15$
1	1713167	$7.9861e-16$

Table 6.1: Numerical results for problem pde900

drastically, while the number of floating point operations increases very modestly. Observe that $\alpha = 1$ corresponds to using the standard Gaussian elimination with partial pivoting.

The following example illustrates how Algorithm 6.1 leads to remarkable increase in the flexibility of the Gauss-Seidel–type of iterations. With fast solvers for k-Hessenberg linear systems we are no more constrained to split the matrix into a lower and an upper triangular part.

Example 6.5 We take A to be the matrix **Sherman1** from the Matrix Market collection. Decompose the matrix as

$$A = D - E - F, \tag{6.52}$$

with D being the band matrix with upper and lower bandwidths k having the entries of A in this band. Moreover, $-E$ is the strictly k-lower part, i.e., $e_{i,j} = 0$, when $j > i - k - 1$ holds and $-F = A - D + E$ is the remaining strictly k-upper part of A.

As is usual in analyzing the Gauss-Seidel iterations, we study $G = I - (D - E)^{-1}A$ and plot the spectrum of G with $k = 0, 1$ and 10. The results are shown in Figure 6.3.

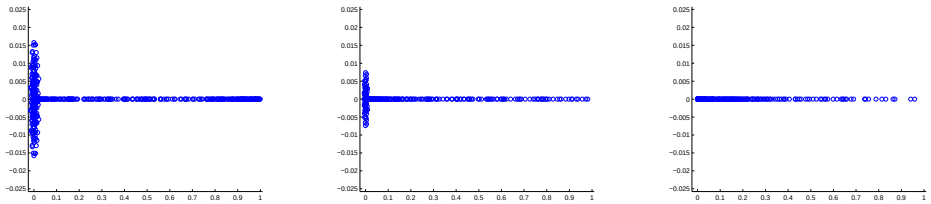


Figure 6.3: The spectrum of G with $k = 0$ (left), $k = 1$ (middle) and $k = 10$ (right) from Example 6.5

This generalization of the Gauss-Seidel splitting has the effect of decreasing the spectral radius $\rho(G)$ and pushing the eigenvalues of G towards the

origin. As a rule of thumb, for a Krylov-method this clustering of eigenvalues can be expected to speed up the convergence significantly [105].

In the following example, we consider the incomplete LH-decomposition as a preconditioner for an iterative method.

Example 6.6 We take A to be the matrix **Sherman4** from the Matrix Market-collection having the size $n = 1104$ with $\mathrm{nz}(A) = 3786$ nonzero entries. We consider solving the right-preconditioned linear system (1.22), i.e.,

$$APy = b, \quad x = Py,$$

with the restarted GMRES method, restarted after the size of the subspace has reached 11. As a right-hand side we use $b = \hat{b}/||\hat{b}||_2$, where $\hat{b} = \mathbf{1} = (1, 1, \ldots, 1)$ denotes a vector of all ones. We use the stopping criterion (5.9), i.e.,

$$\frac{||b - Ax||_2}{||b||_2} \leq 1E - 6,$$

i.e., the iteration is terminated once the relative residual norm of the original linear system (1.1) has been reduced by six orders of magnitude.

As a preconditioner, we use an incomplete LH-decomposition of the downward shifted matrix QA, computed with Algorithm 6.2. Then we have

$$QA \approx LH,$$

where the permutation Q is chosen as in (6.48) and H is an upper k-Hessenberg matrix with a lower bandwidth $k = 0, 25$ and 50. Similarly to the $ILUT$ preconditioner [103], we use numerical dropping by threshold τ and count p, i.e., we store at most the p largest elements of each row of L and H, respectively. The last k columns of the matrix H are not subject to any numerical dropping. We choose $\tau = 1E-1$ and $p = 5$ as the parameters of the numerical dropping. Algorithm 6.1 is used to solve linear systems with the computed incomplete factor H without pivoting, i.e., the pivoting threshold is chosen as $\alpha = 0$.

Denote by $\mathrm{nz}(X)$ the number of nonzero entries in a matrix X. Table 6.2 describes the number of nonzero entries in the computed incomplete factors L and H and the number of matrix-vector products required by the restarted GMRES method with ILH preconditioner to reach the stopping criterion (5.9) for different values of k.

By the results of Table 6.2, we observe that increasing the parameter k improves convergence for the **Sherman4** test problem. Even though the computed factors are very sparse, computational and memory cost of the

Problem	k	nz(L)+nz(H)	Iters
Sherman4	0	5963	274
	25	5972	258
	50	6004	243

Table 6.2: Number of matrix-vector products required to have (5.9) for GMRES(11) with $ILH(1E-1,5)$ preconditioning for the linear system (1.1)

solution of the linear systems involving the factor H increases linearly in k as the parameter k is increased. Therefore, for actual practical computations the values used in this example for k may be too large.

6.7 Conclusions

In this chapter we have presented a fast algorithm for solving sparse k-Hessenberg linear systems. The algorithm has a floating point complexity $O(C(k+1)n)$. The memory consumption is fixed of the order $O(kn)$. In the ideal case for sparse systems, the algorithm is able to circumvent the growth of the number of floating point operations due to fill-in of the standard Gaussian elimination.

The method is matrix-free in the sense that it only requires knowing the entries of H locally. Therefore it does not require the matrix H to be stored in memory.

Because the algorithm is very fast for solving linear systems involving sparse k-Hessenberg matrices, preconditioning general linear systems with k-Hessenberg matrices or by using respective incomplete decompositions is a relevant option. Also k-Hessenberg splittings lead to immediate generalizations of the Gauss-Seidel type of iterations with more flexibility.

Chapter 7

Approximate factoring of the inverse

In this chapter, we consider a right-preconditioned linear system (1.22), written as

$$APy = b, \quad x = Py,$$

where $A \in \mathbb{C}^{n\times n}$, $b \in \mathbb{C}^n$ and P is a preconditioner of an approximate inverse-type.

Define a matrix subspace to be sparse if all its elements are sparse with a common sparsity pattern. Let $\mathcal{W}$ and $\mathcal{V}_1$ to be sparse matrix subspaces of $\mathbb{C}^{n\times n}$ containing invertible elements. In addition let the nonsingular elements of $\mathcal{V}_1$ be readily invertible. As in Chapter 2, we deem a matrix to be readily invertible if solving a linear system with it is inexpensive in terms of floating point operations. We present an algorithm for computing preconditioners of the form

$$P = WV_1^{-1}, \tag{7.1}$$

where $W \in \mathcal{W}$ and $V_1 \in \mathcal{V}_1$.

To obtain a preconditioner P, we approximately factor the inverse of a matrix A into the product WV_1^{-1}. To this end, we consider the problem

$$AW \approx V_1, \tag{7.2}$$

where both W and V_1 are regarded as variables. In terms of matrix subspaces $\mathcal{W}$ and $\mathcal{V}_1$, we formulate the factorization problem by inspecting the nullspace of the linear map $L : \mathcal{W} \longmapsto \mathbb{C}^{n\times n}$, defined as

$$W \longmapsto LW = (I - P_1)AW, \ \text{ with } \ W \in \mathcal{W}, \tag{7.3}$$

where P_1 denotes an orthogonal projection onto $\mathcal{V}_1$ [68].

For a connection between (7.3) and the factorization problem (7.2), assume there exists an exact factorization

$$AWV_1^{-1} = I,$$

with $V_1 = P_1AW$. Then, we have

$$AW = V_1 = P_1AW \iff (I - P_1)AW = \mathbf{0},$$

where $\mathbf{0}$ denotes a matrix of all-zeroes. Thus a factorization exists if and only if there are invertible elements W in the nullspace. Therefore, to have approximate factorization of the form WV_1^{-1}, we consider approximate solutions to

$$\inf_{W\in\mathcal{W}, V_1\in\mathcal{V}_1} ||AWV_1^{-1} - I||, \tag{7.4}$$

in some suitable norm. Here, since the minimum does not need to exist, the infimum is used.

For preconditioning a linear system (1.1) the exact factorization $A = WV_1^{-1}$ is not needed and in practice it would often not be even feasible to compute. In contrast to (7.4), for approximate factorization (7.2) we have $AW \approx V_1 = P_1AW$, if and only if $(I-P_1)AW \approx \mathbf{0}$. This yields a computable minimization problem

$$\min_{W\in\mathcal{W}, ||W||_F=1} ||(I - P_1)AW||_F \tag{7.5}$$

for generating factors W and V_1 in terms of the singular values of the linear map (7.3). In the following sections, we first transform the minimization problem (7.5) into an extremal eigenvalue problem and then present an algorithm based on the power method for computing the factors W and V_1. Assuming $\mathcal{W}$ and $\mathcal{V}_1$ are sparse, by using the power method with sparse-sparse operations we aim to achieve a computational complexity of order $O(\mathrm{nz}(A))$ which is completely free of inner products.

Our factored minimization approach differs from standard approaches to approximate inverses. For a connection between the minimization problem (7.5) and minimization problems usually taken in approximate inverse techniques, consider setting $\mathcal{V}_1 = \mathbb{C}\mathcal{I}$. We then have the standard sparse approximate inverse minimization problem, written as

$$\min_{W\in\mathcal{W}} ||I - AW||_F. \tag{7.6}$$

The factored approximate inverse minimization problem (7.5) can therefore be considered as a generalization of the standard approximate inverse minimization problem (7.6). We note that the minimization problem (7.6) is used quite universally, for instance, in [25] and many others.

In general, the inverse of a sparse matrix can be full. Thus, solving (7.6) with reasonable accuracy may require the dimension of $\mathcal{W}$ to be too large for practical computations. On the other hand, for (7.5), even though the factors are sparse, a factored approximate inverse WV_1^{-1} can be nearly full. For instance, even a simple choice such as taking $\mathcal{V}_1$ to be the subspace of block diagonal matrices with a block size k, we have

$$\mathrm{nz}(WV_1^{-1}) \approx k\mathrm{nz}(W),$$

where $\mathrm{nz}(W)$ denotes the number of nonzero elements in matrix W. We note that in the context of approximate inverses, a block diagonal approach has been used in [75], for instance.

For a relation between (7.4) and (7.5), we have

$$\begin{aligned} ||AW - V_1||_2 &= ||(AWV_1^{-1} - I)V_1||_2 \\ &\leq ||(AWV_1^{-1} - I)||_2||V_1||_2, \end{aligned}$$

and

$$\begin{aligned} ||AW - V_1||_2 &= ||AW - V_1||_2 \frac{||V_1^{-1}||_2}{||V_1^{-1}||_2} \\ &\geq \frac{||(AWV_1^{-1} - I)V_1V_1^{-1}||_2}{||V_1^{-1}||_2} \\ &= \frac{||AWV_1^{-1} - I||_2}{||V_1^{-1}||_2}. \end{aligned}$$

From these we have, after multiplication by $||V_1^{-1}||_2$,

$$||AWV_1^{-1} - I||_2 \leq ||AW - V_1||_2||V_1^{-1}||_2 \leq ||(AWV_1^{-1} - I)||_2\kappa(V_1), \quad (7.7)$$

where $\kappa(V_1) = ||V_1||_2||V_1^{-1}||_2$ denotes the condition number of V_1. Thus, the two approximation problems are related by the conditioning of V_1.

In the next section, an algorithm for computing approximate factors W and V_1 based on the power method is presented. After presenting the algorithm, we address the critical choice of subspaces $\mathcal{W}$ and $\mathcal{V}_1$. Finally, we consider implementation of the method and present numerical experiments where we assess the quality of the computed preconditioners.

7.1 An algorithm for computing approximate factors

To compute the factors W and V_1, a numerical solution of the minimization problem (7.5) is needed. For solving the standard minimization problem

(7.6) in the case where A is nonsymmetric, several algorithms have been proposed. These are based on Minimal Residual (MR) algorithm on the operator $W \longmapsto I - AW$ [25], MR algorithm on the linear systems $Aw_j = e_j$, $j = 1, \ldots, n$, $W = [w_1\ w_2\ \cdots\ w_n]$ [25], optimal techniques for the selection of the nonzero entries of W [58, 66], biorthogonalization techniques [10] and many others, see [105, Chapter 10] and references therein.

Our approach to solving the minimization problem (7.5) is to numerically approximate the smallest singular values of the linear operator (7.3). The singular values of the linear map

$$X \longmapsto AX$$

are determined by A and can be recovered with the help of the Kronecker product $A \otimes I$. Note that with the minimization problem (7.5) we are using a restriction on $\mathcal{W}$ followed by a projection to the orthogonal complement of $\mathcal{V}_1$. A viable alternative for the numerical approximation is the power method, since it is completely free of inner products and requires only a modest amount of additional storage.

For the power method, adjoint of the linear operator L is needed. Denote by $\mathcal{V}$ any matrix subspace $\mathcal{V}$ of $\mathbb{C}^{n\times n}$ over $\mathbb{C}$. We use the trace inner product

$$(M, N) = \operatorname{tr}(N^* M),$$

where $M, N \in \mathcal{V}$. We have the following proposition.

Proposition 7.1. *Adjoint operator of the linear operator L, defined as in (7.3), $L^* : \mathbb{C}^{n\times n} \longmapsto \mathcal{W}$, is*

$$L^* = P_{\mathcal{W}} A^* (I - P_1) : \mathbb{C}^{n\times n} \to \mathcal{W},$$

where $P_{\mathcal{W}}$ denotes orthogonal projection on $\mathbb{C}^{n\times n}$ onto $\mathcal{W}$.

Proof. First consider linear map $W \longmapsto AW$. For $W \in \mathcal{W}$, $X \in \mathbb{C}^{n\times n}$ we have

$$(AW, X) = \operatorname{tr}(X^* AW) = \operatorname{tr}((A^* X)^* W) = \operatorname{tr}(W, P_{\mathcal{W}} A^* X),$$

thus the adjoint is $X \longmapsto P_{\mathcal{W}} A^* X$. For the other part, i.e., linear map $W \longmapsto P_1 AW$, we have

$$\begin{aligned} (P_1 AW, X) &= \operatorname{tr}(X^* P_1 AW) \\ &= \operatorname{tr}((P_1 X)^* AW) \\ &= \operatorname{tr}((A^* P_1 X)^* W) = (W, P_{\mathcal{W}} A^* P_1 X), \end{aligned}$$

and the adjoint is thus $X \longmapsto P_{\mathcal{W}} A^* P_1 X$. By combining the adjoints of the linear maps $W \longmapsto AW$ and $W \longmapsto P_1 AW$, the claim then immediately follows. □

To compute the singular values, we numerically approximate the smallest eigenpairs of the linear map

$$W \longmapsto L^*LW = P_{\mathcal{W}}A^*(I - P_1)AW, \tag{7.8}$$

on the matrix subspace $\mathcal{W}$. Our interest lies in the smallest eigenvalues of (7.8). To approximate the smallest eigenvalues, we consider

$$\alpha I - L^*L,$$

with $\alpha \in \mathbb{R}^+$ chosen in such a way that it makes the smallest eigenvalue of (7.8) dominant. Setting $\alpha = r||A||_2$ with $1/2 < r \leq 3/4$ seems to be a reasonable choice. It is sufficient to have only a rough estimate of $||A||_2$, which is achievable by using a few steps of the power method, for instance. With these, we then have Algorithm 7.1.

Algorithm 7.1 Power method for solving (7.5)

1: Set $\alpha \in \mathbb{R}^+$ and choose an initial factor $W \in \mathcal{W}$
2: **repeat**
3: $\quad M = (I - P_1)AW$
4: $\quad N = P_{\mathcal{W}}A^*M$
5: $\quad W := \alpha W - N$
6: $\quad W := W/||W||_F$
7: **until** stopped

Once Algorithm 7.1 is stopped, we compute $V_1 = P_1AW$ to have the approximate factorization of A as WV_1^{-1}.

We note that the linear map (7.8) is a positive semidefinite operator, so other algorithms than the power method can be used. Many of these, conjugate gradient approaches for instance, will most likely obtain better convergence, but in turn also require more memory. Also, since the resulting W and V_1 will be used for preconditioning, it is not necessary to compute the smallest eigenpair exactly. Being sufficiently close to a cluster is enough for obtaining reasonable convergence rate for a Krylov subspace method, such as GMRES, applied to the preconditioned system. In addition, the power method has the benefit of requiring no inner products, thus being very efficient when parallelized.

We have not yet considered how subspaces $\mathcal{W}$ and $\mathcal{V}_1$ should be chosen. As with the standard approximate inverse techniques, the quality of the preconditioner depends on the structure of the subspaces and how they have been chosen. In the next section, we address the choice of $\mathcal{W}$ and $\mathcal{V}_1$ and present an adaptive way of constructing them via numerical dropping.

7.2 Choice of subspaces $\mathcal{W}$ and $\mathcal{V}_1$

Algorithm 7.1 attempts to solve minimization problem (7.5) in subspaces $\mathcal{W}$ and $\mathcal{V}_1$. Thus the achievable minimum and the quality of the computed preconditioner depends considerably on the structure of these subspaces. Analogously, in traditional techniques dealing with the minimization problem (7.6), the structure of $\mathcal{W}$ has a significant effect on the quality of the preconditioner [24].

We define a matrix subspace $\mathcal{V}_1$ of $\mathbb{C}^{n\times n}$ to be nonsingular if it contains invertible elements. The set of inverses of invertible elements of $\mathcal{V}_1$ is described by

$$\mathrm{Inv}(\mathcal{V}_1) = \{V_1^{-1} |\ V_1 \in \mathcal{V}_1 \cap \mathrm{GL}(n,\mathbb{C})\},$$

where $\mathrm{GL}(n,\mathbb{C})$ denotes the group of invertible matrices in $\mathbb{C}^{n\times n}$ [69]. Recall that we deem elements of $\mathrm{Inv}(\mathcal{V}_1)$ to be readily invertible if operations on the inverses of V_1 to a vector are inexpensive to compute in terms of floating point operations.

Let $A \in \mathbb{C}^{n\times n}$ be nonsingular. With the approximation problem (7.4) for computing approximate factors, we consider the set

$$A\mathcal{W}\mathrm{Inv}(\mathcal{V}_1) = \{AWV_1^{-1} \mid W \in \mathcal{W},\ V_1 \in \mathcal{V}_1 \cap \mathrm{GL}(n,\mathbb{C})\},$$

which does not need to be closed. We have the following example regarding the infimum.

Example 7.1 Let $A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. We choose $\mathcal{W}$ and $\mathcal{V}_1$ to be the spaces of lower and upper triangular matrices in $\mathbb{C}^{2\times 2}$, respectively.

By setting $W = \begin{bmatrix} 1 & 0 \\ 1/\epsilon & 1 \end{bmatrix}$ and $V_1^{-1} = \begin{bmatrix} \epsilon & 1 \\ 0 & (\epsilon^2-1)/\epsilon \end{bmatrix}$ we observe that although non-attainable, for the approximation problem (7.4) the infimum is zero when $\epsilon \mapsto 0$. On the other hand, $I \notin A\mathcal{W}\mathrm{Inv}(\mathcal{V}_1)$.

In the following section we define standard subspaces, which are probably the most common type of subspaces.

7.2.1 Standard subspaces

We have not yet defined how subspaces $\mathcal{W}$ and $\mathcal{V}_1$ are selected when the minimization problem (7.5) is solved. In standard techniques concerned with the minimization problem (7.6), it is common to set subspace $\mathcal{W}$ only to contain entries at certain positions of a matrix. Similar formulation can also be used with factored approximate inverses.

Definition 7.2. *Standard basis matrix of $\mathbb{C}^{n\times n}$ is a matrix containing 1 in exactly one position, and zero otherwise.*

Standard basis matrices can also be considered vectors in $\mathbb{C}^{n^2}$. Assuming columnwise indexing, there exists a bijection f which maps the index (i,j) of an n-by-n array to an index of a vector of size n^2. Denote by $\mathrm{I}_n = \{j \mid 1 \leq j \leq n\}$ the set of indices between 1 and n. For the bijection f, we then have

$$f : \mathrm{I}_n \times \mathrm{I}_n \longmapsto \mathrm{I}_{n^2},\ f(i,j) = (j-1)n + i,$$

with the inverse

$$f^{-1} : \mathrm{I}_{n^2} \longmapsto \mathrm{I}_n \times \mathrm{I}_n,\ f^{-1}(k) = (k - n(\lceil k/n \rceil - 1), \lceil k/n \rceil)\,.$$

With standard basis matrices we now define standard matrix subspaces.

Definition 7.3. *A subspace $\mathcal{V}$ of $\mathbb{C}^{n\times n}$ is called standard if it has a basis consisting of standard basis matrices.*

An equivalent definition of defining a standard subspace is to let the sparsity pattern of $\mathcal{V}$ to determine its dimension.

Using standard subspaces has the benefit of making Algorithm 7.1 readily parallelizable. In addition, with standard subspaces, projection operators P_1 and $P_{\mathcal{W}}$ become straightforward to implement.

Lemma 7.4. *Denote by P the orthogonal projection of matrix on $\mathbb{C}^{n\times n}$ onto a standard subspace $\mathcal{V}$. The projection P is equivalent to selecting the elements in the sparsity pattern of $\mathcal{V}$. Equivalently, for the orthogonal projection $PX = Y \in \mathcal{V}$ we then have*

$$PX = Y, \quad Y_{i,j} = X_{i,j},\ \text{if } V_{i,j} \neq 0,$$

where $V_{i,j} \in \mathcal{V}$ denotes a non-zero entry at position (i,j) in the sparsity pattern of $\mathcal{V}$.

Proof. Consider the orthogonal splitting of X by its sparsity pattern, $X = X_{\mathcal{V}} + X_{\mathcal{V}^\perp}$, where $X_{\mathcal{V}} \in \mathcal{V}$ belongs to the sparsity pattern and $X_{\mathcal{V}^\perp} \perp \mathcal{V}$ into its orthogonal complement. By the definition of an orthogonal projection, we have

$$\begin{aligned} ||X - PX||_F^2 &= \min_{Y\in\mathcal{V}} ||X - Y||_F^2 \\ &= \min_{Y\in\mathcal{V}} ||X_{\mathcal{V}} + X_{\mathcal{V}^\perp} - Y||_F^2 \\ &= \min_{Y\in\mathcal{V}} ||X_{\mathcal{V}} - Y||_F^2 + ||X_{\mathcal{V}^\perp}||_F^2, \end{aligned}$$

which is clearly minimized when $X_{\mathcal{V}} = Y$, i.e., the elements which belong to the sparsity pattern of $\mathcal{V}$ and are nonzero in Y are equal to those in X. □

Thus the implementation of orthogonal projections with standard subspaces is extremely simple. To compute orthogonal projection of $X \in \mathbb{C}^{n\times n}$ onto $\mathcal{V}$, we select exactly those elements of X which are in the sparsity pattern of $\mathcal{V}$.

With standard subspaces, in addition to being simple to implement, projectors P_1 and $P_{\mathcal{W}}$ decouple into operations over the columns (or rows) of $\mathbb{C}^{n\times n}$. Denote by w_j the jth column of W. Also, denote by $P_{1,j}$, $(I-P_1)_j$ and $P_{\mathcal{W},j}$ those parts of the orthogonal projection operators P_1, $(I-P_1)$ and $P_{\mathcal{W}}$ which affect the jth column. Then, the minimization problem (7.5) can be written as

$$\min_{W\in\mathcal{W},||W||_F=1} ||(I-P_1)AW||_F = \min_{w_j\in\mathcal{W}_j,||W||_F=1} \sum_{j=1}^{n} ||(I-P_1)_j Aw_j||_2, \quad (7.9)$$

where $\mathcal{W}_j$ denotes jth column of $\mathcal{W}$. Using equation (7.9), Algorithm 7.1 becomes readily parallelizable. We then have the parallel version of Algorithm 7.1 as Algorithm 7.2 below.

Algorithm 7.2 Parallel power method for computing approximate factors

Let $\alpha \in \mathbb{R}^+$ and $W = [w_1, \ldots, w_n] \in \mathcal{W}$
repeat
  for all columns $j = 1, \ldots, n$ **in parallel do**
    $m_j = (I-P_1)_j Aw_j$
    $n_j := P_{\mathcal{W},j} A^* m_j$
    $w_j := \alpha w_j - n_j$
  end for
  $W := W/||W||_F$
until stopped
Compute $V_1 = P_1 AW$

Regarding projections and standard subspaces, consider computing $m_j = (I-P_1)_j Aw_j$. To have m_j, only those elements in the orthogonal complement of the jth column of $\mathcal{V}_1$ have to be computed. Thus m_j can be regarded as sparsification of Aw_j. Similarly, to have $n_j := P_{\mathcal{W},j} A^* m_j$, only elements of the jth column of $\mathcal{W}$ have to be computed. This can produce significant computational savings, especially if the products Aw_j and A^*m_j are much less sparse than A and $\mathcal{W}$.

In the next two sections, we present some ways of choosing sparsity structures for $\mathcal{W}$ and $\mathcal{V}_1$ when standard subspaces are used.

7.2.2 Standard subspaces from matrix Krylov subspaces

Sparsity patterns for standard subspaces can be constructed from matrix Krylov subspaces of $\mathbb{C}^{n\times n}$, defined as

$$\mathcal{K}_j(A; I) = \{I, A, A^2, \ldots, A^{j-1}\}$$

for $j \leq \deg(A)$, where $\deg(A)$ denotes the degree of A. With $j = \deg(A)$ this contains the inverse of A. Even though $\mathcal{K}_j(A; I)$ is not a standard matrix subspace as such, its sparsity pattern can be used to define sparsity structures for $\mathcal{W}$. By imposing an additional constraint that the elements must be readily invertible, we can use $\mathcal{K}_{j+1}(A; I)$ to obtain a sparsity pattern for $\mathcal{V}_1$.

While constructing the sparsity patterns, we can use sparsification, i.e., numerical dropping of small elements, when the powers of A are computed. A similar technique has been used by Chow in [24] to obtain sparsity patterns for standard approximate inverses.

In this subsection we have considered the construction of sparsity patterns for standard subspaces $\mathcal{W}$ and $\mathcal{V}_1$ a-priori. Another option is to let Algorithm 7.1 construct sparsity structures adaptively.

7.2.3 Standard subspaces via numerical dropping

If there are no sparsity structures for standard subspaces $\mathcal{W}$ and $\mathcal{V}_1$ to use, an option is to construct the structures numerically during the execution of Algorithm 7.1. In each round of the iteration, we then aim to maintain sparsity of $\mathcal{W}$ and $\mathcal{V}_1$ by replacing with zeroes those elements which satisfy certain predefined conditions. With this kind of adaptive construction, the sparsity structures of $\mathcal{W}$ and $\mathcal{V}_1$ evolve during the iteration which can lead to structures that greatly differ from the sparsity structures such as those of A, A^* and A^*A.

Using adaptive numerical techniques for maintaining sparsity of the preconditioner is by no means new. Similar ideas are widely used with standard approximate inverse techniques in selecting sparsity structures for $\mathcal{W}$, see for example [10, 25, 58].

For maintaining sparsity, we adopt the technique of numerical dropping, i.e., replacing with zeroes those entries that meet predefined conditions. Denote by $v \in \mathbb{C}^n$ a vector with entries v_j. To select the relatively large entries of v numerically, we drop entries by relative tolerance τ and by count p. The dropping rules are as follows.

1. Entry v_j is dropped if $|v_j| \leq \tau_j = \tau||v||_2$ holds.

2. Entry v_j is dropped if $\#v_L > p$, where $\#v_L$ denotes the number of entries $v_l, l = 1, \ldots, n, l \neq j$ for which $|v_l| > |v_j|$ holds.

The first rule states that we only store those entries of v that are relatively large. The second rule states that we only store p largest entries of v at most. Note that for finding the relatively small entries of v the direction of the inequality in the first rule and the direction of the second inequality in the second rule are reversed.

We now describe a strategy for adaptively constructing $\mathcal{W}$ and $\mathcal{V}_1$ by using numerical dropping. We set $W = I$ and $\mathcal{W} = \mathcal{I}$ initially. Consider first the line 3 of Algorithm 7.1 and computing $M = (I - P_1)AW$. We define subspace $\mathcal{V}_1$ and projection P_1 by choosing $\mathcal{V}_1$ to contain those entries of AW which are within some predefined readily invertible sparsity pattern and are numerically large. Then, with subspace $\mathcal{V}_1$ and projection P_1 defined, M is acquired simply by dropping those entries of AW which are in P_1.

Similarly, on line 4 for computing $N = P_{\mathcal{W}}A^*M$, we need a sparsity structure for subspace $\mathcal{W}$. To have $\mathcal{W}$ and $P_{\mathcal{W}}$, we locate those entries of A^*M which are numerically large and augment (or redefine) $\mathcal{W}$ to contain the locations of those entries. Then $N = P_{\mathcal{W}}A^*M$ is acquired by dropping those elements of A^*M that are not in $P_{\mathcal{W}}$.

If the subspace $\mathcal{W}$ is fully redefined on line 4, the computed matrix N is likely to be from a different subspace than the original W. Therefore, on line 5, we apply numerical dropping to the computed $W = \alpha W - N$ and update the subspace $\mathcal{W}$ accordingly. With numerical dropping, we have Algorithm 7.3.

Algorithm 7.3 Parallel power method with numerical dropping

1: Let $\alpha \in \mathbb{R}^+$ and $W = [w_1, \ldots, w_n] \in \mathcal{W}$
2: **repeat**
3: **for all** columns $j = 1, \ldots, n$ **in parallel do**
4: Find large entries of Aw_j, update readily invertible $\mathcal{V}_1$
5: $m_j = (I - P_1)_j Aw_j$
6: Find large entries of $(I - P_{\mathcal{W}})_j A^* m_j$, update $\mathcal{W}$
7: $n_j := P_{\mathcal{W},j} A^* m_j$
8: $w_j := \alpha w_j - n_j$
9: Apply numerical dropping to w_j and update $\mathcal{W}$
10: **end for**
11: $W := W/||W||_F$
12: **until** stopped
13: Compute $V_1 = P_1 AW$ and apply numerical dropping to V_1

When numerical dropping is used, the sparsity patterns of $\mathcal{W}$ and $\mathcal{V}_1$ evolve during the course of the iteration. Even though the construction of $\mathcal{V}_1$ is confined to readily invertible matrices, we have the additional restriction that $V_1 \in \mathcal{V}_1$ must be invertible. For arbitrary subspaces $\mathcal{V}_1$, this can be difficult to ensure in practice. For certain subspaces, such as subspaces of block diagonal matrices, invertibility of the elements is easier to verify.

Another difficulty in the construction of $\mathcal{W}$ and $\mathcal{V}_1$ is the added complexity in terms of floating point operations. To be able to update $\mathcal{W}$ and $\mathcal{V}_1$, additional elements from both AW and A^*M must be computed. This creates an additional cost both in memory and in floating point operations.

In the next section, we focus more on the implementation and complexity of Algorithm 7.1.

7.3 Implementation and complexity

For Algorithm 7.1 to be feasible in terms of computational complexity in practice, matrix-matrix products with two sparse matrices need to be computed efficiently.

Consider the product

$$Z = XY,$$

where $X \in \mathbb{C}^{n\times k}$, $Y \in \mathbb{C}^{k\times m}$ and $Z \in \mathbb{C}^{n\times m}$ are sparse matrices. We have

$$z_{ij} = \sum_{l=1}^{k} x_{il} y_{lj},$$

where x_{ij}, y_{ij} and z_{ij} denote the elements at the ith row and jth column of X,Y and Z. Because X and Y are sparse, most of the products $x_{il}y_{lj}$ have either one or both of the factors equal to zero. The computation of Z would then require approximately $O(kmn)$ operations, amount which is too great for any practical computations with large matrices. Testing if either of the factors equal zero explicitly does not suffice, since the operation count would essentially remain the same.

To reduce complexity, matrix-matrix products must be performed in *sparse-sparse*-mode [36, 59, 105], which will be introduced in what follows. Denote by x_j, y_j and z_j the jth columns of X, Y and Z, respectively. By formulating the matrix-matrix product as

$$z_j = \sum_{l=1}^{k} x_l y_{lj}, \tag{7.10}$$

computations only involve operations on sparse column vectors x_j with elements of sparse columns y_j. When formula (7.10) is used, vectors z_j are accumulated as the sum of columns of X multiplied by elements of a single sparse column vector of Y. Therefore this formulation is natural when both X and Y are stored by columns.

For the operation count of (7.10), let $k_1, k_2 \ll n$ and denote by $\mathrm{nz}(X) = O(k_1 k)$ and $\mathrm{nz}(Y) = O(k_2 m)$ the number of nonzero elements in matrices X and Y, respectively. The columns y_j of matrix Y are sparse vectors with k_2 nonzero elements on an average. Due to the sparsity of y_j, only k_2 out of the k products in total are required to compute z_j, which yields only $O(k_1 k_2)$ operations due to the sparsity of X. Thus, the computation of matrix Z requires

$$\sum_{j=1}^{m} \sum_{l=1}^{k} x_l y_{lj} = O(k_1 k_2 m),$$

operations in total.

Note that Algorithm 7.1 requires computation of products with A and A^*. Assuming that A and W have been stored columnwise, computation of the product AW is efficient in terms of floating point operations by using formulation of (7.10). On the other hand, since A^* is available only by rows, the efficient computation would A^*M require M to be stored rowwise, which contradicts previous assumptions. In the computation of A^*M, formula (7.10) may be fully benefited from by storing a columnwise copy of A^*. Then the cost is increased memory usage of the algorithm. Note that it is not absolutely necessary to store A^* explicitly, since computing the product $P_{\mathcal{W}} A^* M$ only requires computing those elements of A^*M which are in the sparsity pattern of the subspace $\mathcal{W}$.

When standard subspaces are used, an operation count for a single iteration of Algorithm 7.1 follows with little discomfort. With standard subspaces, the operation count is determined by the sparsity of A, $\mathcal{W}$ and $\mathcal{V}_1$. For the analysis, we assume the number of nonzero elements to equal $\mathrm{nz}(A) = O(k_1 n)$, $\mathrm{nz}(\mathcal{W}) = O(k_2 n)$ and $\mathrm{nz}(\mathcal{V}_1) = O(k_3 n)$, with $k_1, k_2, k_3 \ll n$.

We first consider line 3 and computing $M = (I - P_1)AW$. The range of projection $(I - P_1)$ is the orthogonal complement of sparse subspace $\mathcal{V}_1$ and is nearly full. The projection operator $(I - P_1)$ results in no significant computational savings. Hence the computational cost for line 3 is approximately

$$O((I - P_1)AW) = O(k_1 k_2 n)$$

operations, which is essentially the cost of computing the product AW in sparse-sparse mode.

On line 4, to have $N = P_{\mathcal{W}} A^* M$, only the elements in the sparse subspace $\mathcal{W}$ have to be computed. Then, the sparsity of A^* ensures that the computational cost for line 4 is

$$O(P_{\mathcal{W}} AM) = O(k_1 k_2 n)$$

operations even when M is full. For line 5

$$O(\alpha W - N) = O(2k_2 n)$$

operations and for line 6

$$O(W/||W||_F) = O(3k_2 n)$$

operations are needed. Combining the complexities, we have the total cost for one iteration round of Algorithm 7.1 as $O((2k_1 k_2 + 5k_2)n)$.

Memory consumption of Algorithm 7.1 seems at first glance quite large, since at least theoretically intermediate product matrix M can be full. Due to the fact that columns m_j may be computed and then discarded after use, the algorithm is feasible for practical computations with large matrices. Thus, during the iteration it is sufficient to store $W \in \mathcal{W}$, A^* and a single vector for storing m_j and n_j, which yields a minimum memory consumption of $O((k_1 + k_2 + 1)n)$ when iteration is done columnwise. Note that $W \in \mathcal{W}$ does not become denser each iteration step, a problem which is a common for standard approximate inverse techniques [105, Chapter 10.5].

We note that the complexity of Algorithm 7.3 is somewhat harder to estimate accurately, since the sparsity structure of $\mathcal{W}$ is not known beforehand. If no heuristics are used, adaptive construction of $\mathcal{W}$ and $\mathcal{V}_1$ requires computing all elements of AW and A^*M, which somewhat increases the complexity of the algorithm. To avoid $O(n^2)$ complexity, the computations must be performed in sparse-sparse mode, which means that A^* must be explicitly stored columnwise.

In the next section, we briefly discuss the prospects of using Algorithm 7.1 in improving a preconditioner.

7.3.1 Improving a preconditioner

Typically, the quality of the preconditioner computed with Algorithm 7.1 depends on the number of iterations that have been taken. Now consider the case that a preconditioner $W \in \mathcal{W}$ has been computed with standard approximate inverse techniques, i.e., by solving the minimization problem (7.6). If the preconditioning performance of W is poor, Algorithm 7.1 can

be used to improve $W \in \mathcal{W}$ by using the existing preconditioner as an initial guess for W and by setting the sparsity structure of $\mathcal{W}$ to equal the sparsity structure of W.

Using Algorithm 7.1 also offers an option to compute a factor $V_1 \in \mathcal{V}_1$ with $\mathcal{V}_1$ something else than the subspace of identity matrices. We then have an approximate factorization of the form

$$AW \approx V_1,$$

with $W \in \mathcal{W}$ and $V_1 \in \mathcal{V}_1$. The idea of improving a preconditioner is similar to idea given by Saad in [105, Chapter 10.5.8]. The main difference is that with Algorithm 7.1 *both* $W \in \mathcal{W}$ *and* $V_1 \in \mathcal{V}_1$ are being refined simultaneously. For another approach to improve an existing preconditioner, see [66, 67].

Choosing a good structure for the subspace $\mathcal{V}_1$ is a difficult task when an existing preconditioner is improved. In terms of the minimization problem (7.5), a greedy approach is to initially define $\mathcal{V}_1$ to contain the largest entries of AW within some readily invertible pattern. Then Algorithm 7.1 and numerical dropping are used to evolve sparsity structure of $\mathcal{V}_1$ as iteration proceeds.

The main concern with every preconditioner is how it performs in practice. In the following section, we present some results on the performance of factored approximate inverses, computed with Algorithms 7.1 and 7.3.

7.4 Numerical experiments

The contents of this section closely follow the contents of [20, Section 5.2]. We illustrate the behaviour of Algorithms 7.1 and 7.3 with numerical experiments performed with MATLAB, version 7.9.0.529 (R2009b). We are concerned with right-preconditioned linear system (1.22) as

$$AMy = b, \quad x = My,$$

with $A \in \mathbb{R}^{n \times n}$. For comparison, the preconditioner M is generated with SPAI-algorithm [58] and with Algorithm 7.3. As SPAI-implementation, we used the generic implementation available online[1] used via MATLAB's mex-interface. As a linear solver we use our own implementation of GMRES(k), GMRES restarted after every k steps. The right-hand side is always $b = \hat{b}/||\hat{b}||_2$, where $\hat{b} = (1, 1, \ldots, 1)$ is the vector of all ones. As a stopping criterion

$$\frac{||b - Ax||_2}{||b||_2} \leq 1E-6$$

[1] `http://www.computational.unibas.ch/software/spai/`

A	n	$\mathrm{nz}(A)$	$\kappa^{\mathrm{est}}(A)$
jpwh911	991	6027	7.3E+2
orsreg1	2205	14133	1E+2
sherman1	1000	3750	2.3E+4

Table 7.1: For Example 7.2, n is the order, $\mathrm{nz}(A)$ the number of nonzero elements and $\kappa^{\mathrm{est}}(A)$ the condition number estimate of A.

is used, i.e., the iteration is terminated once the relative residual is reduced by six orders of magnitude.

Excluding Example 7.3, all the matrices A in our experiments are taken from Matrix Market collection [92].

In our numerical examples, we first examine how quickly Algorithm 7.1 produces a good preconditioner and how the quality of the computed preconditioner scales on a simple model problem. We then show how the sparsity structure of $\mathcal{W}$ evolves when it is automatically determined by using numerical dropping in Algorithm 7.3. Finally, we present a few numerical examples about the convergence behaviour of the restarted GMRES when preconditioned with Algorithm 7.3 for matrices known to be difficult to successfully precondition with approximate inverse techniques.

In the following numerical example, we consider how the number of refinements with Algorithm 7.1 affects the quality of the preconditioner. This can also be considered as improving a preconditioner $M = I$ with $\mathcal{V}_1$ chosen as the set of diagonal matrices.

Example 7.2 Here the purpose is to illustrate how many iterates are typically needed with Algorithm 7.1 to produce reasonable approximate factors.

We experiment with three matrices A by solving the preconditioned linear system (1.22) with GMRES(20). The matrices sherman1 and orsreg1 arise in oil reservoir simulation and jpwh991 in circuit simulation. Some of their properties are listed in Table 7.1.

We denote by W_k and V_1 the factors obtained after performing k iterates with Algorithm 7.1 to improve an initial guess W_0. We use somewhat artificial choices for matrix subspaces. To have a symmetric sparsity pattern for AW, we take $\mathcal{W}$ to be the standard subspaces having the nonzero pattern of A^*, while $\mathcal{V}_1$ is the set of diagonal matrices. For simplicity, we take $W_0 = I$ as an initial guess.

From Table 7.2 we observe that for the selected problems, the quality of the preconditioner does not improve much after 10 iterations. This indicates

Matrix / M	I	$W_5V_1^{-1}$	$W_{10}V_1^{-1}$	$W_{15}V_1^{-1}$	$W_{20}V_1^{-1}$	$W_{40}V_1^{-1}$
jpwh991	53	26	24	23	23	22
sherman1	> 1000	720	470	292	176	129
orsreg1	303	198	174	179	178	175

Table 7.2: The number of steps GMRES(20) required for jpwh911, sherman1 and orsreg1: unpreconditioned (I) and preconditioned using different number of iterations with Algorithm 7.1.

that the power iteration to obtain the preconditioner has converged to a cluster of eigenvalues in the left end of the spectrum of the linear map (7.3). In other numerical experiments we have observed similar behaviour, with some dependence on the initial guess W_0 and the sparsity structure of $\mathcal{W}$. The sherman1 matrix was an exception in the sense that preconditioner kept improving. This was most likely caused by an unfavourable structure of the spectrum of (7.8) causing W_k to converge relatively slowly.

In the following example, we consider the quality of the preconditioner computed with Algorithm 7.1 in relation to the size of a simple model problem.

Example 7.3 We consider an elliptic model problem in three dimensions, i.e., the three dimensional Poisson problem,

$$\begin{cases} -\Delta u &= f, \quad \text{in } \Omega \\ u &= 0, \quad \text{on } \partial\Omega \end{cases}, \tag{7.11}$$

on a unit cube $\Omega = (0,1)^3$ with $f = 1$. We discretize the problem (7.11) with the standard second order seven point finite difference stencil by using $(N+1)$ grid points in each direction. We then have the grid size as $h = 1/N$. The discretization yields a linear system (1.1) with a Hermitian positive definite matrix $A \in \mathbb{R}^{n\times n}$ with $n = N^3$. The matrix A has a block tridiagonal structure with N blocks of the size N^2-by-N^2 in each direction. Denote by I_k a k-by-k identity matrix. We then have

$$A = \frac{1}{h^2}\begin{bmatrix} Y & -I_{N^2} & & & & & \\ -I_{N^2} & Y & -I_{N^2} & & & & \\ & -I & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & -I_{N^2} & \\ & & & & -I_{N^2} & Y & -I_{N^2} \\ & & & & & -I_{N^2} & Y \end{bmatrix},$$

where the N^2-by-N^2 matrix Y is a block tridiagonal matrix with N-by-N blocks. We have

$$Y = \begin{bmatrix} T & -I_N & & & & & \\ -I_N & T & -I_N & & & & \\ & -I_N & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & -I_N & \\ & & & & -I_N & T & -I_N \\ & & & & & -I_N & T \end{bmatrix},$$

where the N-by-N matrix T is tridiagonal with the diagonal entries all equal to 6 and the entries in the first sub- and superdiagonals all equal to -1.

We take $\mathcal{W}$ to be a standard subspace having the nonzero pattern of A and set $W_0 = I$ as an initial guess. The sparsity pattern of the standard subspace $\mathcal{V}_1$ was chosen to be the set of block diagonal matrices with tridiagonal blocks with a blocksize of N^2, i.e., $\mathcal{V}_1$ contains N tridiagonal blocks in total. To construct a preconditioner, we take 15 iterations with Algorithm 7.1.

Table 7.3 describes the number of matrix-vector products required by the restarted GMRES method without and with factored approximate inverse preconditioning. Nonconvergence before the limit of 2000 iterations is denoted by dagger (†). For each grid size h, we also describe the number of nonzero entries in the matrix A and in the computed approximate factors W and V_1. Note that since V_1 is tridiagonal, the computational cost of an LU-decomposition is approximately $O(8N^2)$ operations for each block, i.e., $O(8n)$ operations in total. In addition, the LU-decomposition of V_1 can be computed in place, i.e., no additional storage is required by the computed factors [55].

h	n	nz(A)	nz(W)+nz(V_1)	Iters	Iters prec.
1/8	512	3200	4608	24	13
1/16	4096	27136	38912	92	26
1/32	32768	223232	319488	325	87
1/64	262144	1810432	2588672	1184	267
1/128	2097152	14581760	20840448	†	951

Table 7.3: Number of iterations required for convergence with GMRES(10) with and without factored approximate inverse preconditioning

From the results of Table 7.3 we observe that the number of iterations required by GMRES(10) to reduce the initial residual by six orders of magni-

tude increases as the grid size decreases. However, since halving the grid size increases the size of the problem by 8, the number of iterations required by GMRES(10) preconditioned with a factorized approximate inverse computed with Algorithm 7.1 is approximately $\sim 3^{\log_2(N)-1}$.

We note that as the size of the problem was increased, the quality of the preconditioner became more sensitive to the number of iterations used in Algorithm 7.1. For instance, for the case $h = 1/16$ the number of iterations required by GMRES(10) iterations did not reduce even if the preconditioner was constructed by using 50 iterations of Algorithm 7.1. On the other hand, for the case $h = 1/64$ the number of the required GMRES(10) iterations was reduced to 255 when the preconditioner was constructed by using 50 iterations of Algorithm 7.1.

This behaviour may be due to the distribution of the singular values of the linear map (7.3), which are determined by the eigenvalues of the matrix A. It is well known, see for instance [81], that the eigenvalues of the matrix A are of the form $\lambda_{k,l,m} = (k^2 + l^2 + m^2)\pi^2$, where $k, l, m = 1, \ldots, N$, i.e., we have $\lambda_{N,N,N} \to \infty$ as $N \to \infty$. Then the size of the cluster containing the smallest eigenvalues can be experted to grow as the grid size increases, which may slightly reduce the quality of the computed preconditioner if the number of iteration steps in Algorithm 7.1 is kept constant.

In the remaining experiments we employ Algorithm 7.3. Then it is not necessary to fix the sparsity patterns of $\mathcal{W}$ and $\mathcal{V}_1$ beforehand as they evolve during the iteration. This is certainly a more flexible and user-friendly black box alternative. More importantly, it provides high quality preconditioners that seem to be computable in a small number of iterations, such as five.

Example 7.4 We take the sherman1 matrix and examine the evolution of the sparsity pattern of W after $k = 1, 10, 20, 30$ and 40 iterations with Algorithm 7.3. To keep the initial sparsity pattern as simple as possible, we use $W_0 = I$ as an initial guess. The matrix subspace $\mathcal{V}_1$ is the set of diagonal matrices.

We set $\tau = 1E - 05$ and $p = 30$ as parameters for numerical dropping. With these, the evolution of the sparsity pattern of W is depicted in Figure 7.1.

Since $W_0 = I$, after one iteration the structure of $\mathcal{W}$ resembles the nonzero structure of A^*A. As the iteration progresses, the nonzero structure of $\mathcal{W}$ becomes more unpredictable and begins to differ from the sparsity patterns of A^*, A or A^*A. This indicates that an artificially defined sparsity pattern for $\mathcal{W}$ (and $\mathcal{V}_1$) may not be sufficient in capturing significant entries of A^{-1}. See [58] for similar conclusions made with the SPAI algorithm.

In practice we do not take $\mathcal{V}_1$ to be the set of diagonal matrices, it would

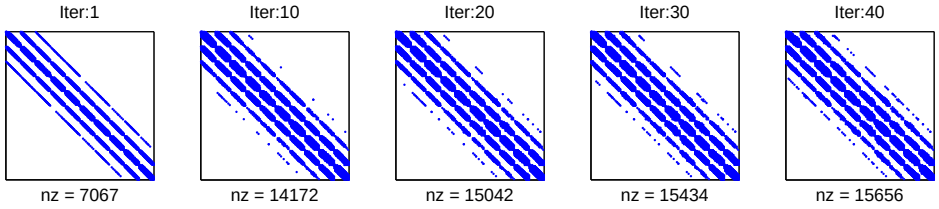


Figure 7.1: For Example 7.4, the evolution of the sparsity pattern of W for the sherman1 matrix after $k = 1, 10, 20, 30$ and 40 iterations with Algorithm 7.3.

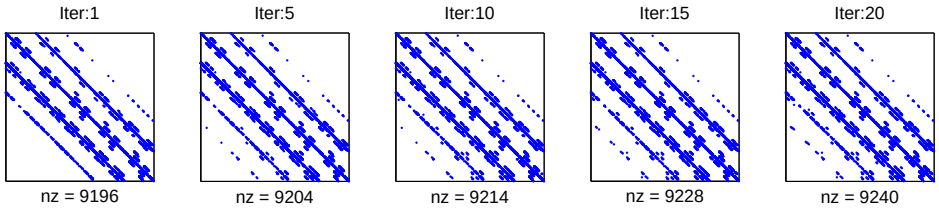


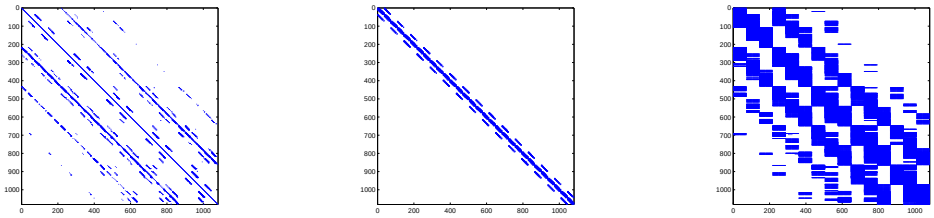
Figure 7.2: For Example 7.5, the evolution of the sparsity pattern of W for the sherman2 matrix after $k = 1, 5, 10, 15$ and 20 iterations with Algorithm 7.3.

be too restrictive. We want to benefit more from the freedom of choosing a suitable matrix subspace containing readily invertible elements. Even simple alternatives, like choosing subspace $\mathcal{V}_1$ as the space of block diagonal matrices can be surprisingly effective in solving challenging problems.

Example 7.5 We take A to be the sherman2 matrix which is known to be difficult to solve with iterative methods without preconditioning and proper scaling, especially with approximate inverse techniques [8]. Then $n = 1080$ with nz(A)= 23094 and a condition number estimate $\kappa^{\text{est}}(A) = 1.4E + 12$.

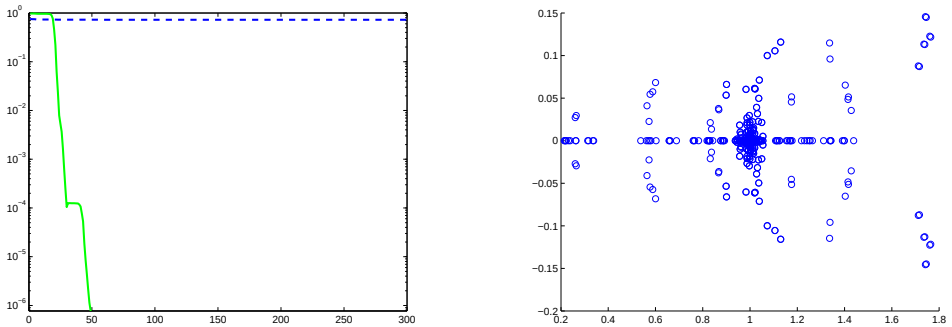
With Algorithm 7.3, the sparsity pattern of $\mathcal{W}$ was determined by numerical dropping with the parameters $\tau = 1E - 10$ and $p = 10$. An initial guess $W_0 = I$ was used. The matrix subspace $\mathcal{V}_1$ was chosen to be the set of block diagonal matrices with a block size 72. Numerical dropping with the parameters $\tau = 1E - 10$ and $p = 15$ was used to keep V_1 sparse.

The evolution of the sparsity pattern of W after $1, 5, 10, 15$ and 20 iterations with Algorithm 7.3 is depicted in Figure 7.2.



(a) W (b) V_1 (c) WV_1^{-1}

Figure 7.3: For Example 7.5, the sparsity patterns of the factors W and V_1 and the preconditioner WV_1^{-1}.

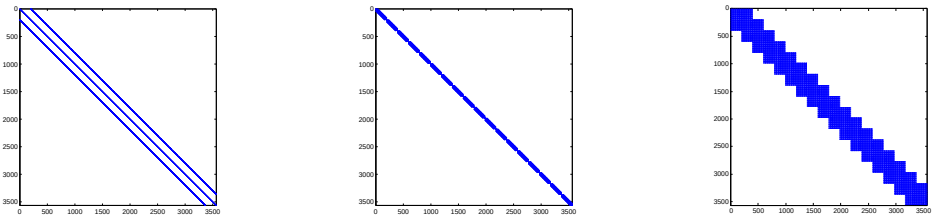


(a) (b)

Figure 7.4: (a) Convergence of GMRES(30) with SPAI(0.3) (dashed line) and Algorithm 7.3 (solid line). (b) The spectrum of the preconditioned matrix AWV_1^{-1}.

Figure 7.3 shows the sparsity patterns of W, V_1 and the respective preconditioner WV_1^{-1} after just 5 iterations. The number of nonzero elements were nz(W)= 9204, nz(V_1)= 12190 and nz(WV_1^{-1})= 198837. (Of course, WV_1^{-1} was computed only to obtain its sparsity structure and the number of nonzero elements.) Using this, we solve the preconditioned linear system (1.22) with GMRES(30). For comparison, we also constructed a preconditioner M with SPAI with nz(M) = 21336. The convergence results are shown in Figure 7.4(a). Observe that with the preconditioner of Algorithm 7.3, we are able to solve the problem in about 50 iterations.

As shown in Figure 7.4(b), the spectrum of the preconditioned operator is well clustered around 1.



(a) W (b) V_1 (c) WV_1^{-1}

Figure 7.5: For Example 7.6, the sparsity patterns of the factors W and V_1 and the preconditioner WV_1^{-1}.

Example 7.6 We take A to be the saylor4 matrix arising in petroleum engineering simulation. Then $n = 3564$ with nz(A)= 22316 and a condition number estimate $\kappa(A) = 1E + 2$. In [58] it was reported that when SPAI is used to precondition the problem, GMRES(20) does not converge within 1000 iterations.

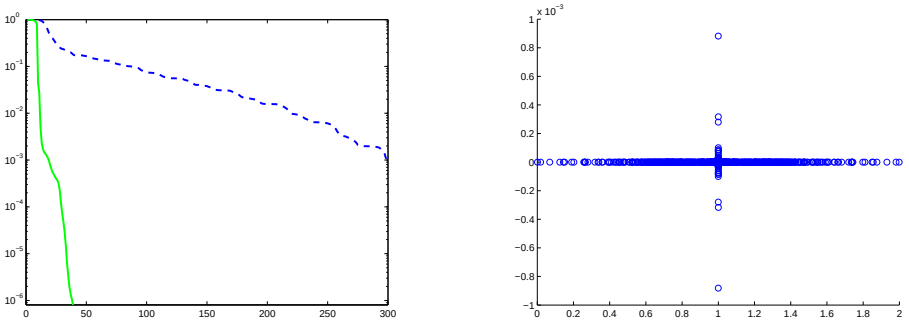
The sparsity pattern of $\mathcal{W}$ was determined by Algorithm 7.3 with the numerical dropping parameters $\tau = 1E - 5$ and $p = 10$. An initial guess $W_0 = I$ was used. The matrix subspace $\mathcal{V}_1$ was chosen to be the set of block diagonal matrices with a block size 198. Numerical dropping with the parameters $\tau = 1E - 5$ and $p = 20$ was used to keep V_1 sparse.

To construct a preconditioner, we take 5 iterations with Algorithm 7.3. The sparsity patterns for W, V_1 and WV_1^{-1} are depicted in Figure 7.5. The number of nonzero elements were nz(W)=26482, nz(V_1)=15626 and nz(WV_1^{-1})=2009304. Using this, we solve the preconditioned linear system (1.22) with GMRES(20). For comparison, we also constructed a preconditioner M with SPAI with nz(M)= 202343. The convergence results are shown in Figure 7.6(a). With Algorithm 7.3, we managed to solve the problem in 39 iterations. We were not able to solve the problem with the SPAI preconditioner in spite of using considerably more fill-in.

As shown in Figure 7.6(b), the spectrum of the preconditioned operator is well clustered around 1.

Let us mention that in Examples 7.5 and 7.6 Algorithm 7.3 was somewhat sensitive to the block size of $\mathcal{V}_1$. This dependence is apparently related with the way the problems have been discretized. Unless this information is available, it might be a good idea to use band matrices instead.

In the following examples, it is shown that Algorithm 7.3 can be used in preconditioning Hermitian problems as well.



(a) (b)

Figure 7.6: (a) Convergence of GMRES(20) with SPAI(0.1) using 10 refinements and 10 new non-zeroes per refinement (dashed line) and Algorithm 7.3 (solid line). (b) The spectrum of the preconditioned matrix AWV_1^{-1}.

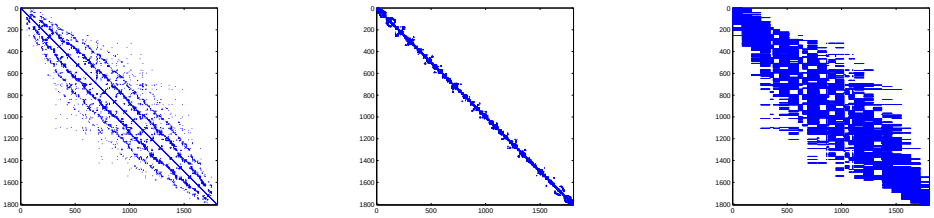
Example 7.7 We take A the bcsstk14 matrix which is Hermitian and positive definite. Then $n = 1806$ with nz(A)= 32630 and a condition number estimate of $\kappa(A) = 1.3E + 10$.

The sparsity pattern of $\mathcal{W}$ was determined by Algorithm 7.3 with the numerical dropping parameters $\tau = 1E - 4$ and $p = 20$. An initial guess $W_0 = I$ was used. The matrix subspace $\mathcal{V}_1$ was chosen to be the set of block diagonal matrices with a block size 86. Numerical dropping with the parameters $\tau = 1E - 10$ and $p = 20$ was used to keep V_1 sparse

To construct a preconditioner, we take 5 iterations with Algorithm 7.3. The sparsity patterns for W, V_1 and WV_1^{-1} are depicted in Figure 7.7. The number of nonzero elements were nz(W)=10052, nz(V_1)=29460 and nz(WV_1^{-1})=307560. Using this, we solve the preconditioned linear system (1.22) with GMRES(30). For comparison, we also constructed a preconditioner M with SPAI with nz(M)=93823. The convergence results are shown in Figure 7.8(a). With Algorithm 7.3, we managed to solve the problem in 245 iterations. With SPAI, we were not able to solve the problem within 1000 iterations.

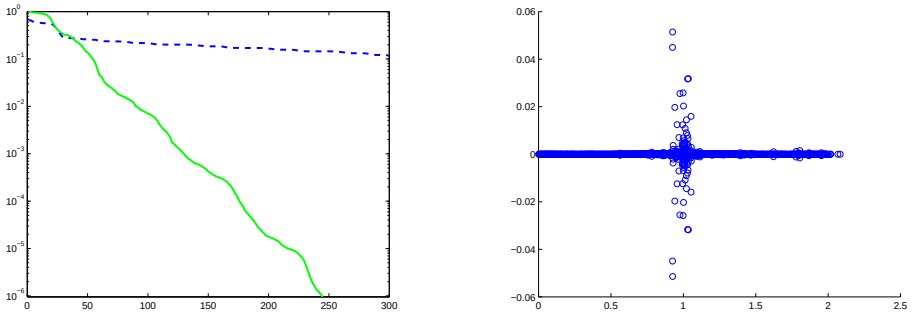
The spectrum of AWV_1^{-1} is clustered around the point 1. This clustering of the eigenvalues seems to yield satisfactory speed of convergence allowing us to solve the problem with GMRES(30) in a reasonable number of iterations.

Let us emphasize that in Examples 7.3 and 7.7 we did not use the fact that the matrix was Hermitian positive definite. As a rule, in preconditioning, the structure should be benefited from for faster convergence. Also, since usually one wants to have a symmetric linear system also after the preconditioner



(a) W (b) V_1 (c) WV_1^{-1}

Figure 7.7: For Example 7.7, the sparsity patterns of the factors W and V_1 and the preconditioner WV_1^{-1}.



(a) (b)

Figure 7.8: (a) Convergence of GMRES(30) with SPAI(0.1) using 10 refinements and 10 new non-zeroes per refinement (dashed line) and Algorithm 7.3 (solid line). (b) The spectrum of the preconditioned matrix AWV_1^{-1}.

has been applied, subspaces $\mathcal{W}$ and $\mathcal{V}_1$ as well as Algorithm 7.1 should be modified accordingly.

7.5 Conclusions

In this chapter, we have proposed a new method for computing approximate factors for the inverse of a large and sparse matrix. The method is based on converting a minimization problem into a hermitian eigenproblem, which is approximately solved with the power method with a reasonable number of iterations. Computation of the approximate factorization is executed in sparse-sparse mode for complexity and storage reasons.

In [9], a comparison between SPAI and the approximate factorization AINV of [10] is presented, with the conclusion that the performance of the algorithms is largely similar. The preliminary numerical results of this chapter show that with Algorithm 7.3, the performance of SPAI may be significantly improved.

Currently the main problem with Algorithm 7.1 is the choice of subspaces $\mathcal{W}$ and $\mathcal{V}_1$. Some possibilities for choosing $\mathcal{W}$ and $\mathcal{V}_1$ were given, but in general the choice of subspaces remains largely an open problem. When the choice of subspaces becomes more clear, a thorough comparison between Algorithm 7.3 with SPAI and other approximate inverse algorithms should be conducted.

Bibliography

[1] E. Agullo, A. Guermouche, and J.-Y. L'Excellent, *A parallel out-of-core multifrontal method: storage of factors on disk and analysis of models for an out-of-core active memory*, Parallel Comput., 34 (2008), pp. 296–317.

[2] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster, *A fully asynchronous multifrontal solver using distributed dynamic scheduling*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 15–41 (electronic).

[3] W. E. Arnoldi, *The principle of minimized iteration in the solution of the matrix eigenvalue problem*, Quart. Appl. Math., 9 (1951), pp. 17–29.

[4] Z. Bai and Y. Su, *SOAR: a second-order Arnoldi method for the solution of the quadratic eigenvalue problem*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 640–659 (electronic).

[5] B. Beckermann and L. Reichel, *Error estimates and evaluation of matrix functions via the Faber transform*, SIAM J. Numer. Anal., 47 (2009), pp. 3849–3883.

[6] M. Bellalij, Y. Saad, and H. Sadok, *Further analysis of the Arnoldi process for eigenvalue problems*, SIAM J. Numer. Anal., 48 (2010), pp. 393–407.

[7] M. Benzi, *Preconditioning techniques for large linear systems: a survey*, J. Comput. Phys., 182 (2002), pp. 418–477.

[8] M. Benzi, J. C. Haws, and M. Tůma, *Preconditioning highly indefinite and nonsymmetric matrices*, SIAM J. Sci. Comput., 22 (2000), pp. 1333–1353 (electronic).

[9] M. Benzi and M. Tůma, *Numerical experiments with two approximate inverse preconditioners*, BIT, 38 (1998), pp. 234–241.

[10] ——, *A sparse approximate inverse preconditioner for nonsymmetric linear systems*, SIAM J. Sci. Comput., 19 (1998), pp. 968–994 (electronic).

[11] ——, *A parallel solver for large-scale Markov chains*, Appl. Numer. Math., 41 (2002), pp. 135–153. Developments and trends in iterative methods for large systems of equations—in memoriam Rüdiger Weiss (Lausanne, 2000).

[12] D. Bindel, J. Demmel, W. Kahan, and O. Marques, *On computing Givens rotations reliably and efficiently*, ACM Trans. Math. Software, 28 (2002), pp. 206–238.

[13] D. S. Bindel, J. W. Demmel, W. Kahan, and O. A. Marques, *On computing givens rotations reliably and efficiently*, Tech. Rep. 148, LAPACK Working Note, Oct. 2000.

[14] Å. Björck, *Solving linear least squares problems by Gram-Schmidt orthogonalization*, Nordisk Tidskr. Informations-Behandling, 7 (1967), pp. 1–21.

[15] ——, *Numerics of Gram-Schmidt orthogonalization*, Linear Algebra Appl., 197/198 (1994), pp. 297–316. Second Conference of the International Linear Algebra Society (ILAS) (Lisbon, 1992).

[16] M. Bollhöfer, *A robust and efficient ILU that incorporates the growth of the inverse triangular factors*, SIAM J. Sci. Comput., 25 (2003), pp. 86–103 (electronic).

[17] M. Bollhöfer and Y. Saad, *ILUPACK - preconditioning software package, release v1.0, may 14, 2004*. `http://www.tu-berlin.de/ilupack/`.

[18] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A multigrid tutorial*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second ed., 2000.

[19] R. Bru, J. Marín, J. Mas, and M. Tůma, *Balanced incomplete factorization*, SIAM J. Sci. Comput., 30 (2008), pp. 2302–2318.

[20] M. Byckling and M. Huhtanen, *Approximate factoring of the inverse*, Numerische Mathematik, (2010), pp. 1–22. 10.1007/s00211-010-0341-4.

[21] P. Castillo and Y. Saad, *Preconditioning the matrix exponential operator with applications*, J. Sci. Comput., 13 (1998), pp. 275–302.

[22] C. Chevalier and F. Pellegrini, *PT-Scotch: a tool for efficient parallel graph ordering*, Parallel Comput., 34 (2008), pp. 318–331.

[23] E. Chow, *Parallel Sparse Approximate Inverse (Least-Squares) Preconditioner*. `https://computation.llnl.gov/casc/parasails/parasails.html`.

[24] E. Chow, *A priori sparsity patterns for parallel sparse approximate inverse preconditioners*, SIAM J. Sci. Comput., 21 (2000), pp. 1804–1822 (electronic). Iterative methods for solving systems of algebraic equations (Copper Mountain, CO, 1998).

[25] E. Chow and Y. Saad, *Approximate inverse preconditioners via sparse-sparse iterations*, SIAM J. Sci. Comput., 19 (1998), pp. 995–1023 (electronic).

[26] J. W. Daniel, W. B. Gragg, L. Kaufman, and G. W. Stewart, *Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization*, Math. Comp., 30 (1976), pp. 772–795.

[27] B. N. Datta, *Numerical Linear Algebra and Applications, Second Edition*, SIAM-Society for Industrial and Applied Mathematics, 2 ed., January 2010.

[28] T. A. Davis, *Algorithm 832: UMFPACK V4.3—an unsymmetric-pattern multifrontal method*, ACM Trans. Math. Software, 30 (2004), pp. 196–199.

[29] ——, *Direct methods for sparse linear systems*, vol. 2 of Fundamentals of Algorithms, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2006.

[30] T. A. Davis, J. R. Gilbert, S. I. Larimore, and E. G. Ng, *A column approximate minimum degree ordering algorithm*, ACM Trans. Math. Software, 30 (2004), pp. 353–376.

[31] J. W. Demmel, S. C. Eisenstat, J. R. Gilbert, X. S. Li, and J. W. H. Liu, *A supernodal approach to sparse partial pivoting*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 720–755 (electronic).

[32] J. Díaz, J. Petit, and M. Serna, *A survey of graph layout problems*, ACM Comput. Surv., 34 (2002), pp. 313–356.

[33] T. A. Driscoll, K.-C. Toh, and L. N. Trefethen, *From potential theory to matrix iterations in six steps*, SIAM Rev., 40 (1998), pp. 547–578 (electronic).

[34] J. Drkošová, A. Greenbaum, M. Rozložník, and Z. Strakoš, *Numerical stability of GMRES*, BIT, 35 (1995), pp. 309–330.

[35] V. Druskin and L. Knizhnerman, *Extended Krylov subspaces: approximation of the matrix square root and related functions*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 755–771 (electronic).

[36] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct methods for sparse matrices*, Monographs on Numerical Analysis, The Clarendon Press Oxford University Press, New York, second ed., 1989. Oxford Science Publications.

[37] I. S. Duff and J. Koster, *On algorithms for permuting large entries to the diagonal of a sparse matrix*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 973–996 (electronic).

[38] I. S. Duff and G. A. Meurant, *The effect of ordering on preconditioned conjugate gradients*, BIT Numerical Mathematics, 29 (1989), pp. 635–657. 10.1007/BF01932738.

[39] V. Eijkhout, *Automatic Determination of Matrix Blocks*, LAPACK Working Note 151, Department of Computer Science, University of Tennessee, Knoxville, Knoxville, TN 37996, USA, Apr. 2001. UT-CS-01-458, April 2001.

[40] T. Eirola and O. Nevanlinna, *Accelerating with rank-one updates*, Linear Algebra Appl., 121 (1989), pp. 511–520. Linear algebra and applications (Valencia, 1987).

[41] N. S. Ellner and E. l. Wachspress, *New ADI model problem applications*, in Proceedings of 1986 ACM Fall joint computer conference, ACM '86, Los Alamitos, CA, USA, 1986, IEEE Computer Society Press, pp. 528–534.

[42] M. Embree, *The tortoise and the hare restart GMRES*, SIAM Rev., 45 (2003), pp. 259–266 (electronic).

[43] ——, *The Arnoldi eigenvalue iteration with exact shifts can fail*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 1–10.

[44] H. Fassbender, *Symplectic methods for the symplectic eigenproblem*, Kluwer Academic/Plenum Publishers, New York, 2000.

[45] P. Favati, G. Lotti, and O. Menchi, *Non-recursive solution of sparse block Hessenberg systems*, Numer. Linear Algebra Appl., 11 (2004), pp. 391–409.

[46] R. Fletcher, *Conjugate gradient methods for indefinite systems*, in Numerical analysis (Proc 6th Biennial Dundee Conf., Univ. Dundee, Dundee, 1975), Springer, Berlin, 1976, pp. 73–89. Lecture Notes in Math., Vol. 506.

[47] V. Frayssé, L. Giraud, S. Gratton, and J. Langou, *Algorithm 842: a set of GMRES routines for real and complex arithmetics on high performance computers*, ACM Trans. Math. Software, 31 (2005), pp. 228–238.

[48] R. W. Freund, *A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems*, SIAM J. Sci. Comput., 14 (1993), pp. 470–482.

[49] R. W. Freund and N. M. Nachtigal, *QMR: a quasi-minimal residual method for non-Hermitian linear systems*, Numer. Math., 60 (1991), pp. 315–339.

[50] L. Gemignani and G. Lotti, *Efficient and stable solution of M-matrix linear systems of (block) Hessenberg form*, SIAM J. Matrix Anal. Appl., 24 (2003), pp. 852–876 (electronic).

[51] A. George and J. W. H. Liu, *Computer solution of large sparse positive definite systems*, Prentice-Hall Inc., Englewood Cliffs, N.J., 1981. Prentice-Hall Series in Computational Mathematics.

[52] N. E. Gibbs, W. G. Poole, Jr., and P. K. Stockmeyer, *An algorithm for reducing the bandwidth and profile of a sparse matrix*, SIAM J. Numer. Anal., 13 (1976), pp. 236–250.

[53] L. Giraud, J. Langou, M. Rozložník, and J. van den Eshof, *Rounding error analysis of the classical Gram-Schmidt orthogonalization process*, Numer. Math., 101 (2005), pp. 87–100.

[54] G. H. Golub and H. A. van der Vorst, *Closer to the solution: iterative linear solvers*, in The state of the art in numerical analysis (York, 1996), vol. 63 of Inst. Math. Appl. Conf. Ser. New Ser., Oxford Univ. Press, New York, 1997, pp. 63–92.

[55] G. H. Golub and C. F. Van Loan, *Matrix computations*, Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, third ed., 1996.

[56] A. Greenbaum, V. Pták, and Z. Strakoš, *Any nonincreasing convergence curve is possible for GMRES*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 465–469.

[57] L. Grigori, J. W. Demmel, and X. S. Li, *Parallel symbolic factorization for sparse LU with static pivoting*, SIAM J. Sci. Comput., 29 (2007), pp. 1289–1314 (electronic).

[58] M. J. Grote and T. Huckle, *Parallel preconditioning with sparse approximate inverses*, SIAM J. Sci. Comput., 18 (1997), pp. 838–853.

[59] F. G. Gustavson, *Two fast algorithms for sparse matrices: multiplication and permuted transposition*, ACM Trans. Math. Software, 4 (1978), pp. 250–269.

[60] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards, 49 (1952), pp. 409–436 (1953).

[61] N. J. Higham, *Accuracy and stability of numerical algorithms*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second ed., 2002.

[62] M. Hochbruck and C. Lubich, *On Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 34 (1997), pp. 1911–1925.

[63] M. Hochbruck, C. Lubich, and H. Selhofer, *Exponential integrators for large systems of differential equations*, SIAM J. Sci. Comput., 19 (1998), pp. 1552–1574 (electronic).

[64] W. Hoffmann, *Iterative algorithms for Gram-Schmidt orthogonalization*, Computing, 41 (1989), pp. 335–348.

[65] L. Hoffnung, R.-C. Li, and Q. Ye, *Krylov type subspace methods for matrix polynomials*, Linear Algebra Appl., 415 (2006), pp. 52–81.

[66] T. Huckle and A. Kallischko, *Frobenius norm minimization and probing for preconditioning*, Int. J. Comput. Math., 84 (2007), pp. 1225–1248.

[67] T. Huckle, A. Kallischko, A. Roy, M. Sedlacek, and T. Weinzierl, *An efficient parallel implementation of the MSPAI preconditioner*, Parallel Comput., 36 (2010), pp. 273–284.

[68] M. Huhtanen, *Factoring matrices into the product of two matrices*, BIT, 47 (2007), pp. 793–808.

[69] ——, *Matrix subspaces and determinantal hypersurfaces*, Ark. Mat., 48 (2010), pp. 57–77.

[70] M. Huhtanen and R. M. Larsen, *Exclusion and inclusion regions for the eigenvalues of a normal matrix*, SIAM J. Matrix Anal. Appl., 23 (2002), pp. 1070–1091 (electronic).

[71] M. Huhtanen and O. Nevanlinna, *A minimum residual algorithm for solving linear systems*, BIT, 46 (2006), pp. 533–548.

[72] C. G. J. Jacobi, *Über ein leichtes verfahren die in der theorie der säculärstörungen vorkommenden gleichungen numerisch aufzulösen*, Crelle's Journal, 30 (1846), pp. 51–94.

[73] C. Jagels and L. Reichel, *The extended Krylov subspace method and orthogonal Laurent polynomials*, Linear Algebra Appl., 431 (2009), pp. 441–458.

[74] ——, *Recursion relations for the extended Krylov subspace method*, Linear Algebra and its Applications, (2010).

[75] C. Janna, M. Ferronato, and G. Gambolati, *A block fsai-ilu parallel preconditioner for symmetric positive definite linear systems*, SIAM Journal on Scientific Computing, 32 (2010), pp. 2468–2484.

[76] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM J. Sci. Comput., 20 (1998), pp. 359–392 (electronic).

[77] L. Knizhnerman and V. Simoncini, *A new investigation of the extended Krylov subspace method for matrix function evaluations*, Numer. Linear Algebra Appl., 17 (2010), pp. 615–638.

[78] L. A. Knizhnerman, *Calculation of functions of nonsymmetric matrices by means of Arnoldi's method*, Zh. Vychisl. Mat. i Mat. Fiz., 31 (1991), pp. 5–16.

[79] D. Kressner and C. Tobler, *Krylov Subspace Methods for Linear Systems with Tensor Product Structure*, SIAM Journal on Matrix Analysis and Applications, 31 (2010), pp. 1688–1714.

[80] C. Lanczos, *Solution of systems of linear equations by minimized-iterations*, J. Research Nat. Bur. Standards, 49 (1952), pp. 33–53.

[81] S. Larsson and V. Thomée, *Partial differential equations with numerical methods*, vol. 45 of Texts in Applied Mathematics, Springer-Verlag, Berlin, 2003.

[82] N. Li, Y. Saad, and E. Chow, *Crout versions of ILU for general sparse matrices*, SIAM J. Sci. Comput., 25 (2003), pp. 716–728 (electronic).

[83] R.-C. Li and Q. Ye, *A Krylov subspace method for quadratic matrix polynomials with application to constrained least squares problems*, SIAM J. Matrix Anal. Appl., 25 (2003), pp. 405–428 (electronic).

[84] MathWorks, *Matlab.* `http://www.mathworks.com/products/matlab/`.

[85] J. A. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp., 31 (1977), pp. 148–162.

[86] C. Moler and C. Van Loan, *Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later*, SIAM Rev., 45 (2003), pp. 3–49 (electronic).

[87] R. B. Morgan, *A restarted GMRES method augmented with eigenvectors*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 1154–1171.

[88] ——, *Implicitly restarted GMRES and Arnoldi methods for nonsymmetric systems of equations*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1112–1135 (electronic).

[89] ——, *GMRES with deflated restarting*, SIAM J. Sci. Comput., 24 (2002), pp. 20–37 (electronic).

[90] N. M. Nachtigal, S. C. Reddy, and L. N. Trefethen, *How fast are nonsymmetric matrix iterations?*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 778–795. Iterative methods in numerical linear algebra (Copper Mountain, CO, 1990).

[91] O. Nevanlinna, *Convergence of iterations for linear equations*, Lectures in Mathematics ETH Zürich, Birkhäuser Verlag, Basel, 1993.

[92] NIST, *Matrixmarket*. `http://math.nist.gov/MatrixMarket/`.

[93] M. Olschowka and A. Neumaier, *A new pivoting strategy for Gaussian elimination*, Linear Algebra Appl., 240 (1996), pp. 131–151.

[94] C. C. Paige, M. Rozložník, and Z. Strakoš, *Modified Gram-Schmidt (MGS), least squares, and backward stability of MGS-GMRES*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 264–284 (electronic).

[95] B. N. Parlett, D. R. Taylor, and Z. A. Liu, *A look-ahead Lánczos algorithm for unsymmetric matrices*, Math. Comp., 44 (1985), pp. 105–124.

[96] D. W. Peaceman and H. H. Rachford, Jr., *The numerical solution of parabolic and elliptic differential equations*, J. Soc. Indust. Appl. Math., 3 (1955), pp. 28–41.

[97] J. K. Reid, *On the method of conjugate gradients for the solution of large sparse systems of linear equations*, in Large sparse sets of linear equations (Proc. Conf., St. Catherine's Coll., Oxford, 1970), Academic Press, London, 1971, pp. 231–254.

[98] W. Rudin, *Real and complex analysis*, McGraw-Hill Book Co., New York, third ed., 1987.

[99] A. Ruhe, *Numerical aspects of Gram-Schmidt orthogonalization of vectors*, Linear Algebra Appl., 52/53 (1983), pp. 591–601.

[100] Y. Saad, *Analysis of some Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 29 (1992), pp. 209–228.

[101] ——, *Numerical methods for large eigenvalue problems*, Algorithms and Architectures for Advanced Scientific Computing, Manchester University Press, Manchester, 1992.

[102] ——, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Comput., 14 (1993), pp. 461–469.

[103] Y. Saad, *ILUT: a dual threshold incomplete LU factorization*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.

[104] ——, *SPARSKIT: a basic tool kit for sparse matrix computations - version 2*, 1994.

[105] ——, *Iterative methods for sparse linear systems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, second ed., 2003.

[106] Y. Saad and M. H. Schultz, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[107] Y. Saad and H. A. van der Vorst, *Iterative solution of linear systems in the 20th century*, J. Comput. Appl. Math., 123 (2000), pp. 1–33. Numerical analysis 2000, Vol. III. Linear algebra.

[108] Y. Saad and K. Wu, *DQGMRES: a direct quasi-minimal residual algorithm based on incomplete orthogonalization*, Numer. Linear Algebra Appl., 3 (1996), pp. 329–343.

[109] V. Simoncini, *A new iterative method for solving large-scale Lyapunov matrix equations*, SIAM J. Sci. Comput., 29 (2007), pp. 1268–1288.

[110] V. Simoncini and D. B. Szyld, *Recent computational developments in Krylov subspace methods for linear systems*, Numer. Linear Algebra Appl., 14 (2007), pp. 1–59.

[111] G. Starke, *Optimal alternating direction implicit parameters for nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 28 (1991), pp. 1431–1445.

[112] ——, *Alternating direction preconditioning for nonsymmetric systems of linear equations*, SIAM J. Sci. Comput., 15 (1994), pp. 369–384. Iterative methods in numerical linear algebra (Copper Mountain Resort, CO, 1992).

[113] G. W. Stewart, *On the solution of block Hessenberg systems*, Numer. Linear Algebra Appl., 2 (1995), pp. 287–296.

[114] J. Todd, *Applications of transformation theory: a legacy from Zolotarev (1847–1878)*, in Approximation theory and spline functions (St. John's, Nfld., 1983), vol. 136 of NATO Adv. Sci. Inst. Ser. C Math. Phys. Sci., Reidel, Dordrecht, 1984, pp. 207–245.

[115] Top500, *Top500 supercomputing sites*. `http://www.top500.org/`.

[116] L. N. Trefethen, *Approximation theory and numerical linear algebra*, in Algorithms for approximation, II (Shrivenham, 1988), Chapman and Hall, London, 1990, pp. 336–360.

[117] L. N. Trefethen and D. Bau, III, *Numerical linear algebra*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[118] G. Vainikko, *Fast solvers of the Lippmann-Schwinger equation*, tech. rep., Institute of Mathematics, Helsinki University of Technology, 1997.

[119] J. van den Eshof and M. Hochbruck, *Preconditioning Lanczos approximations to the matrix exponential*, SIAM J. Sci. Comput., 27 (2006), pp. 1438–1457 (electronic).

[120] H. A. van der Vorst, *Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.

[121] ——, *Iterative Krylov methods for large linear systems*, vol. 13 of Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, Cambridge, 2009. Reprint of the 2003 original.

[122] H. A. van der Vorst and G. H. Golub, *150 years old and still alive: eigenproblems*, in The state of the art in numerical analysis (York, 1996), vol. 63 of Inst. Math. Appl. Conf. Ser. New Ser., Oxford Univ. Press, New York, 1997, pp. 93–119.

[123] H. A. van der Vorst and C. Vuik, *GMRESR: a family of nested GMRES methods*, Numer. Linear Algebra Appl., 1 (1994), pp. 369–386.

[124] M. van Veldhuizen, *A note on partial pivoting and Gaussian elimination*, Numerische Mathematik, 29 (1977), pp. 1–10.

[125] R. S. Varga, *Matrix iterative analysis*, Prentice-Hall Inc., Englewood Cliffs, N.J., 1962.

[126] E. L. Wachspress, *Extended application of alternating direction implicit iteration model problem theory*, J. Soc. Indust. Appl. Math., 11 (1963), pp. 994–1016.

[127] ——, *Iterative solution of the Lyapunov matrix equation*, Appl. Math. Lett., 1 (1988), pp. 87–90.

[128] H. F. Walker, *Implementation of the GMRES method using Householder transformations*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 152–163.

[129] J. W. Watts, *A conjugate gradient truncated direct method for the iterative solution of the reservoir simulation pressure equation*, Society of Petroleum Engineer Journal, 21 (1981), pp. 345–353.

[130] D. B. West, *Introduction to graph theory*, Prentice Hall Inc., Upper Saddle River, NJ, 1996.

[131] J. H. Wilkinson, *The evaluation of the zeros of ill-conditioned polynomials. I, II*, Numer. Math., 1 (1959), pp. 150–180.

[132] ——, *Error analysis of direct methods of matrix inversion*, J. Assoc. Comput. Mach., 8 (1961), pp. 281–330.

[133] ——, *The algebraic eigenvalue problem*, Clarendon Press, Oxford, 1965.

[134] ——, *The perfidious polynomial*, in Studies in numerical analysis, vol. 24 of MAA Stud. Math., Math. Assoc. America, Washington, DC, 1984, pp. 1–28.

[135] D. M. Young, *Iterative solution of large linear systems*, Academic Press, New York, 1971.

(continued from the back cover)

A591 Juhana Siljander
Regularity for degenerate nonlinear parabolic partial differential equations
September 2010

A590 Ehsan Azmoodeh
Riemann-Stieltjes integrals with respect to fractional Brownian motion and applications
September 2010

A589 Antti Rasila, Jarno Talponen
Convexity properties of quasihyperbolic balls on Banach spaces
August 2010

A588 Kalle Mikkola
Real solutions to control, approximation, factorization, representation, Hankel and Toeplitz problems
June 2010

A587 Antti Hannukainen, Rolf Stenberg, Martin Vohralík
A unified framework for a posteriori error estimation for the Stokes problem
May 2010

A586 Kui Du, Olavi Nevanlinna
Minimal residual methods for solving a class of R-linear systems of equations
May 2010

A585 Daniel Aalto
Boundedness of maximal operators and oscillation of functions in metric measure spaces
March 2010

A584 Tapio Helin
Discretization and Bayesian modeling in inverse problems and imaging
February 2010

A583 Wolfgang Desch, Stig-Olof Londen
Semilinear stochastic integral equations in L_p
December 2009

HELSINKI UNIVERSITY OF TECHNOLOGY INSTITUTE OF MATHEMATICS
RESEARCH REPORTS

The reports are available at *http://math.tkk.fi/reports/* .
The list of reports is continued inside the back cover.

A596 Olavi Nevanlinna
Multicentric Holomorphic Calculus
November 2010

A595 Juho Könnö, Rolf Stenberg
Numerical computations with H(div)-finite elements for the Brinkman problem
November 2010

A594 Atte Aalto, Jarmo Malinen
Cauchy problems from networks of passive boundary control systems
October 2010

A593 Toni Lassila
Model reduction and level set methods for shape optimization problems
October 2010

A592 Olavi Nevanlinna
Upper bounds for $\mathbb{R}$-linear resolvents
September 2010

ISBN 978-952-60-3532-1 (print)
ISBN 978-952-60-3533-8 (PDF)
ISSN 1797-5867
Aalto University, Mathematics, 2011